Accelerating the Computation and Design of Nanoscale Materials with Deep Learning

by Kevin Ryczko

Thesis submitted to the University of Ottawa in partial Fulfillment of the requirements for the Doctor of Philosophy in Physics

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Kevin Ryczko

December 2, 2021

Acknowledgments

In times like these, I always seem to ask the question: "How did I get here?" My educational journey has been a long one, with plenty of ups and downs along the way. I firstly must acknowledge two professors from Ontario Tech University that initially drew me into physics: Dr. Joseph MacMillan and Dr. Rupinder Brar. I first entered university as an engineering student with little guidance. After sitting through a year of introductory physics lectures from these two professors, I was hooked to learn more. Shortly after, I changed degrees to study physics. This leads me to my second acknowledgment. Thank you to my parents, Les and Victoria Ryczko, who always let me follow my passions. Although graduate school was out of your realm of guidance, you continued to support me, no matter the outcome.

Later on in my undergraduate degree, I met my supervisor, Dr. Isaac Tamblyn. Isaac took me under his wing and provided mentorship and academic guidance that shaped me into the person I am today. I cannot thank you enough for everything you have done for me over the years. From understanding social networks to atomistic systems, to deep learning, you taught me how to critically think in a wide variety of subjects.

During my studies, I also had the opportunity of visiting different institutions. First, my visit to the Argonne National Laboratory where I met some great people, including Dr. Pierre Darancet. Thank you Pierre, for all of your scientific guidance throughout our collaboration. Second, my visit to the Université de Montréal. Thank you to everyone in the group who made me feel extremely welcomed. Thank you to Dr. Michel Côté for your scientific guidance and your willingness to solve any problem we faced. Last, my visit to the National

Research Council of Canada, where I spent most of my time during my Ph.D. studies. Thank you to all of the people who had a direct impact on me, and thank you for being such a lively, open, and friendly bunch.

In the final year of my Ph.D. studies, I applied for an internship at 1QBit. However, during the interview process, the role changed to a permanent one upon graduation. Since then, I have been working there part-time and I have met some great people. I would like to say thank you to everyone for making the experience so far an enjoyable one. I look forward to continuing our work together in the future.

Lastly, I would like to thank my partner, future mother, and best friend, Courtney Clarke. Ever since we met you have always been there for me and I have no doubt that will change in the future. As this chapter closes I cannot wait to see what the next chapter holds.

Abstract

In this article-based thesis, we cover applications of deep learning to different problems in condensed matter physics, where the goal is to either accelerate the computation or design of a nanoscale material. We first motivate and introduce how machine learning methods can be used to accelerate traditional condensed matter physics calculations. In addition, we discuss what designing a material means, and how it has been previously done. We then consider the fundamentals of electronic structure and conventional calculations which include density functional theory (DFT), density functional perturbation theory (DFPT), quantum Monte Carlo (QMC), and electron transport with tight binding. In addition, we cover the basics of deep learning. Afterwards, we discuss 6 articles. The first 5 articles are dedicated to accelerating the computation of nanoscale materials. In Article 1, we use convolutional neural networks to predict energies for diatomic molecules modelled with a Lennard-Jones potential and density functional theory energies of hexagonal lattices with and without defects. In Article 2, we use extensive deep neural networks to represent density functional theory energy functionals for electron gases by using the electron density as input and bypass the Kohn-Sham equations by using the external potential as input. In addition, we use deep convolutional inverse graphics networks to map the external potential directly to the electron density. In Article 3, we use voxel deep neural networks (VDNNs) to map electron densities to kinetic energy densities and functional derivatives of the kinetic energies for graphene lattices. We also use VDNNs to calculate an electron density from a direct minimization calculation and introduce a Monte Carlo based solver that avoids taking a functional derivative altogether. In Article 4, we use a deep learning framework to predict the polarization, dielectric function, Born-effective charges, longitudinal

optical transverse optical splitting, Raman tensors, and Raman spectra for 2 crystalline systems. In Article 5, we use VDNNs to map DFT electron densities to QMC energy densities for graphene systems, and compute the energy barrier associated with forming a Stone-Wales defect. In Article 6, we design a graphene-based quantum transducer that has the ability to physically split valley currents by controlling the pn-doping of the lattice sites. The design is guided by an neural network that operates on a pristine lattice and outputs a lattice with pn-doping such that valley currents are optimally split. Lastly, we summarize the thesis and outline future work.

Statement of Contributions

Some work in this thesis was completed in collaboration, and therefore includes coauthored works. Only first author works are included in this thesis. Some secondary works are discussed, but most have been omitted entirely.

The author has contributed to the following refereed articles over the course of their studies:

1. **Ryczko, K.**, Mills, K., Luchak, I., Homenick, C. and Tamblyn, I., 2018. Convolutional neural networks for atomistic systems. Computational Materials Science, 149, pp.134-142.

2. Mills, K., **Ryczko, K.**, Luchak, I., Domurad, A., Beeler, C. and Tamblyn, I., 2019. Extensive deep neural networks for transferring small scale learning to large scale systems. Chemical Science, 10(15), pp.4129-4140.

3. **Ryczko, K.**, Strubbe, D.A. and Tamblyn, I., 2019. Deep learning and density-functional theory. Physical Review A, 100(2), p.022512.

4. **Ryczko, K.**, Darancet, P. and Tamblyn, I., 2020. Inverse Design of a Graphene-Based Quantum Transducer via Neuroevolution. The Journal of Physical Chemistry C, 124(48), pp.26117-26123.

5. Choubisa, H., Askerka, M., **Ryczko, K.**, Voznyy, O., Mills, K., Tamblyn, I. and Sargent, E.H., 2020. Crystal site feature embedding enables exploration of large chemical spaces. Matter, 3(2), pp.433-448.

6. Wetzel, S.J., **Ryczko, K.**, Melko, R.G. and Tamblyn, I., 2020. Twin Neural Network Regression. arXiv preprint arXiv:2012.14873 (Under review)

7. Feugmo, C.G.T., **Ryczko, K.**, Anand, A., Singh, C.V. and Tamblyn, I., 2021. Neural evolution structure generation: High Entropy Alloys. The Journal of Chemical Physics. 155(4), pp.044102.

8. **Ryczko, K.**, Wetzel, S.J., Melko, R.G. and Tamblyn, I., 2021. Orbital-Free

Density Functional Theory with Small Datasets and Deep Learning. arXiv preprint arXiv:2104.05408. (Submitted)

9. **Ryczko, K.**, Melanfant, O., Côté, M. and Tamblyn, I., 2021. Electronic Response Quantities of Solids and Deep Learning. (Submitted)

10. Melanfant, O., **Ryczko, K.**, Côté, M. and Tamblyn, I., 2021. Training Deep Neural Networks for Derivative Predictions. (In preparation)

11. **Ryczko, K.**, Krogel, J. and Tamblyn, I., 2021. Toward Acceleration of Quantum Monte Carlo with Deep Learning. (In preparation)

Chapter 5 includes discussion and a figure from Mills, K., Ryczko, K., Luchak, I., Domurad, A., Beeler, C. and Tamblyn, I., 2019. Extensive deep neural networks for transferring small scale learning to large scale systems. Chemical Science, 10(15), pp.4129-4140. I contributed to this work, but did not serve as the primary author. The methodology developed in this article was used in Chapter 8.

Chapter 7 is from Ryczko, K., Mills, K., Luchak, I., Homenick, C. and Tamblyn, I., 2018. Convolutional neural networks for atomistic systems. Computational Materials Science, 149, pp.134-142. All authors revised the manuscript. Iryna Luchak performed experiments verifying the work. Kyle Mills contributed text to the introduction. Isaac Tamblyn supervised the project. All remaining work was done by Kevin Ryczko.

Chapter 8 is from Ryczko, K., Strubbe, D.A. and Tamblyn, I., 2019. Deep learning and density-functional theory. Physical Review A, 100(2), p.022512. All authors revised the manuscript. Isaac Tamblyn supervised the project. All work was done by Kevin Ryczko.

Chapter 9 is from Ryczko, K., Wetzel, S.J., Melko, R.G. and Tamblyn, I., 2021. Orbital-Free Density Functional Theory with Small Datasets and Deep Learning. arXiv preprint arXiv:2104.05408. (Submitted). All authors revised the manuscript. Sebastian Wetzel wrote the Monte Carlo code. Isaac Tamblyn supervised the project. All remaining work was done by Kevin Ryczko.

Chapter 10 is from Ryczko, K., Melanfant, O., Côté, M. and Tamblyn, I., 2021. Response Quantities of Solids and Deep Learning. (Submitted). All authors revised the manuscript. Michel Côté and Isaac Tamblyn co-supervised the project. All remaining work was done by Kevin Ryczko.

Chapter 11 is from Ryczko, K., Krogel, J. and Tamblyn, I., 2021. Deep Learning and Quantum Monte Carlo. (In preparation). Jaren Krogel implemented the energy density estimators. Isaac Tamblyn and Kevin Ryczko revised the manuscript. Isaac Tamblyn supervised the project. All remaining work was done by Kevin Ryczko.

Chapter 12 is from Ryczko, K., Darancet, P. and Tamblyn, I., 2020. Inverse Design of a Graphene-Based Quantum Transducer via Neuroevolution. The Journal of Physical Chemistry C, 124(48), pp.26117-26123. All authors revised the manuscript. Part of the introduction was written by Pierre Darancet. Pierre Darancet and Isaac Tamblyn co-supervised the project. The remaining work was done by Kevin Ryczko.

Contents

Part I

Introduction

Chapter 1

Introductory Remarks

1.1 Background

Materials science involves understanding the properties of materials and how these properties change with a perturbation. This understanding is essential to develop cheaper, more efficient, and sustainable technologies for the future. Some technologies include better large-scale batteries for energy storage [1], drug discovery for future pandemics [2], integrated circuits for next generation electronics [3], and other materials for commercial purposes [4], to name a few examples. Upon understanding the characteristics of a material and how they change with a perturbation, one can then make alterations to the material in an intelligent way to achieve a desired outcome. This concept, we refer to as the *inverse design* of a material.

Before designing a material, one must be able to probe the material and extract information from it. This could be in the form of an experiment or a theoretical calculation. Experiments can be expensive, both in the monetary sense as well as time and effort. When constantly making alterations to a material to optimize a given property, it may not be feasible to perform measurements rapidly in a laboratory. However, in theory and computation, alterations can be done via simulations. An initial cost is necessary for code development and computational resources, but simulations can be run in parallel making them less taxing than physical experiments. In this case, the focus is the *computation* of a certain observable. It is the combination of rapid computation and a material proposal mechanism that is needed to accelerate the design of a material. One must accelerate the computation of a material

before accelerating the design.

Modern day computational materials science involves studying materials at the nano- or meso-scale level [5]. This means that we are investigating materials at length scales on the order of 1-100 Å, depending on the level of theory. The more sophisticated the simulation (lower level theory), the more limited one is in simulating large systems. Here, we aim to understand the microscopic properties of a material, which allows us, afterwards, to explain the macroscopic properties observed in an experiment. In order to compare theory with experiment, we want to best replicate the experimental environment within simulations. This drives one to study larger systems, for longer time scales. To give some intuition, a μm^2 of graphene contains 35.2 million atoms. This length scale has been used in a past experiment to image graphene with a scanning tunneling microscope [6].

Levels of theory range from classical simulations, where nuclei obey Newtonian dynamics (electrons dynamics are excluded), to quantum simulations, where both electrons and nuclei obey the Schrödinger equation. In classical simulations, empirical potentials are used to describe ion-ion interactions. These potentials include parameters that are *fit* to either experimental values or lower-level theories. The functional forms for these potentials are simple, and include radial cut-offs when considering particle interactions. This allows one to simulate millions of atoms for long time scales [7]. One could argue that these simple functions, that are fit to a set of data, form a primitive machine learning (ML) *model*. One adjusts parameters within the interaction potential (or model) such that observables of the underlying data are reproduced. The idea of fitting parameters within a model is not a new concept. Let's consider another example. Instead of fitting an inter-atomic, empirical potential, one can also fit matrix elements of a Hamiltonian. This is exactly what is done in tight-binding approximation, where interaction parameters are fit to reproduce results from a lower-level theory. In fully quantum simulations, referred to as *ab initio* simulations, however, there are no parameters. These simulations are exact, but require solving coupled partial differential equations which quickly become too complex to solve.

In this thesis, we mainly focus on a level of theory in between these two regimes, where electrons are treated as quantum mechanical particles, but nuclei are treated classically. The quantum mechanical problem of understanding electrons, and their interactions is called the electronic structure problem. In this regime, the most widely used electronic structure method is density functional theory (DFT), more specifically Kohn-Sham DFT (KS-DFT), which has computational scaling of $\mathcal{O}(N_{\mathrm{e}}^3)$, where N_{e} is the number of electrons [8]. In comparison to other *ab initio* methods, the computational scaling of KS-DFT is favourable. In addition, the accuracy of KS-DFT for certain observables yields satisfactory results when compared to experiment. Therefore, KS-DFT has been widely adopted in various communities including chemistry, physics, and engineering. The ultimate goal is to use KS-DFT (or even lower level theories) for large-scale simulations that recreate experimental conditions. However, typical simulations with KS-DFT are limited to thousands of atoms. Empirical potentials and tight-binding approaches can be used to model KS-DFT calculations with a lower computational cost, but they are limited in accuracy. For simulations with empirical potentials, there may be cases where the potential energy surface cannot be described by a two-body interaction or by a certain choice of functional form [9]. For tight-binding simulations, commonly used atomic basis functions are only valid when the electrons are tightly-bound to the nuclei within the lattice. Another option, which has recently received a lot of attention in the literature, is the use of machine learning models to approximate observables calculated via KS-DFT (or lower level theories) [10]. ML models are universal function approximators with no explicit functional form and are infinitely differentiable with a suitable choice of activation function. They can be used for *any* material and can cover a wide variety of electronic environments, given enough training data. In addition, evaluation of an observable with an ML model, or *inference*, can be many orders of magnitude faster than a traditional calculation.

1.2 Literature Review

One of the first application of ML methods to KS-DFT was done by Behler and Parinello [11], where they used artificial neural networks (ANNs) to represent the potential energy surfaces calculated via KS-DFT. Their approach was similar to the existing, empirical approach where one divides the total energy amongst individual atoms. One of their main contributions was the design of feature vectors for atomistic systems. When designing an input representation for an atomic environment, the representation should reflect the symmetries of that particular environment. This led to the development of symmetric and translational invariant functions to be used when describing an atomic environment for a particular atom. The choice of symmetry functions is not unique, however. In addition, one is free to choose a ML architecture that performs best on a dataset. The search for an optimal input representation and ML architecture has caused an influx of ML works applied to atomistic systems [12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]. In Ref. [12], input representations were built from Clebsch-Gordon coefficients and Gaussian process regression (GPR) was used to fit potential energy surfaces computed with KS-DFT. In short, GPR is a classical machine learning method where a prediction for a particular input depends on the co-variance between the input and all other training inputs. This methodology, however, does not scale well to larger training set sizes. In Ref. [13], DFT potential energy surfaces were also fit using GPR, but with a new representation built from Coulomb matrices. The elements of the Coulomb matrix are defined as

$$M_{ij} = \begin{cases} 0.5Z_i^{2.4} & \text{when } i = j, \\ \frac{Z_i Z_j}{|\boldsymbol{r}_i - \boldsymbol{r}_j|} & \text{when } i \neq j, \end{cases} \tag{1.1}$$

where Z_i and $\boldsymbol{r}_i$ are the charge and position of nuclei i. A similar approach was taken in Ref. [24], where energy differences between Hartree-Fock theory and coupled cluster singles, doubles, and selected triples (CCSD(T)) were learned using Coulomb matrices and kernel ridge regression (KRR). In Hartree-Fock theory, the wavefunction is written as a Slater determinant of electron orbitals with a particular basis, and the basis coefficients are found self-consistently. A caveat of Hartree-Fock theory is that electron correlation

is absent. In CCSD(T), the wavefunction is not written from a single Slater determinant, but many Slater determinants where electrons are virtually excited, and therefore occupy higher lying electron orbitals. This increase in wavefunction complexity allows for the inclusion of correlation energy but significantly increases the amount of computation that must be performed. Following this work, Ref. [16] introduced a graph neural network (GNN) architecture where information can be passed from nearest neighbours. In this architecture, nuclei are embedded based on their atomic number, and the embedding is updated based on the embedding of the neighbouring atoms and the distances between them. This methodology was later reformulated and presented as SchNet [25]. Since the release of SchNet, other GNN-based architectures have been developed. This includes DimeNet [23] and OrbNet [21]. The underlying machine learning ideology among these architectures are similar. The only difference is the information that is propagated between nearest neighbour atoms when updating atomic embeddings. In addition, all of these works focus on model comparisons for DFT calculations of molecules (i.e. the QM9 dataset [26]) where all molecules are in their equilibrium geometries. More recently, however, a GNN architecture, called a crystal graph convolutional neural network, was used to reproduce calculations (most within chemical accuracy) of crystals using data from the Materials Project [27]. In Ref. [28], a neural network was used to predict charge densities and density of states for Aluminum slabs. There is no unique way of applying a machine learning method to condensed matter systems.

In other works, the focus of ML has been on representing energy functionals to be used in DFT calculations. As described in Section 2.2.2, representing the exact kinetic energy functional with an ML model would improve the accuracy of existing calculations. Furthermore, orbital-free DFT (OF-DFT) could also benefit from more accurate kinetic energy functionals. In Ref. [14], a kinetic energy functional was learned using electron densities as input to KRR for a 1d model system of non-interacting electrons. Derivatives were then taken of the ML model, which initially had large errors. This model system was revisited in [29], where the derivatives were also included in the loss function, which improved the errors. In Ref. [30], it was shown that convolutional neural networks (CNNs) can

be used to map electron densities to accurate exchange-correlation energies computed via CCSD(T) calculations. In this case, the systems were small molecules consisting of H and He.

The majority of this thesis is dedicated to accelerating the computation of nanoscale systems. Before continuing, we summarize and discuss the research direction that this thesis covers. First, a majority of the literature that use ML models are for molecular systems (mainly organic molecules). Second, some ML models require *feature engineering*, which incorporate the correct physics into the models, but puts limits on the machine learning architecture that can be used. Deep learning (DL) techniques have received a lot of attention due to advances with image and video processing and are now being rapidly adopted in the physical sciences. These techniques work on raw data, and learn optimal features during training. The physics can be learned rather than input into the models. With this mentality, we can benefit from more sophisticated, general purpose techniques that are being developed in the ML community. This is the route that is taken in this thesis. We study how DL algorithms can be applied to study condensed matter systems. In addition, in each study we aim to address the limitations and improve on these limitations in a further study.

The remainder of this thesis is dedicated to the design of nanoscale devices. As discussed previously, the acceleration of design first requires the acceleration of the computation. However, even with accelerated computation, the design of a structure is a daunting task. Often one finds that the design space for a particular structure is overwhelmingly large, and a brute force, or high throughput strategy is not sufficient. As eluded to in Ref. [31], inverse design problems have been overcome in the past when mapped to a solvable optimization problems. This includes the self-consistent field cycle in conventional KS-DFT calculations or a structural relaxation of an atomistic system. However, it may be difficult to map an arbitrary design problem to an optimization problem. Typically, one is left with two strategies when it comes to inverse design. One strategy is the use of stochastic methods with some sort of evolutionary algorithm. Another strategy is to utilize the gradients of an objective function. However, to apply this strategy there must exist a direct relationship between the objective function (which represents how optimal a structure is) and the

optimization variables. In the case of structural optimization, for example, the energy is directly related to the atomic coordinates, which allows one to define gradients. There may be cases, however, where there is no (obvious) link between the optimization function and optimization variables. In this case, one can randomly sample the configuration space in parallel, and keep only the best performing structures. The optimal structures can then be mutated or mixed together to form new ones. This is known as an evolutionary strategy and examples optimizing different quantities of materials are given in Refs. [32, 33]. A novel strategy, which has recently come to fruition in the machine learning community, is the use of a generative machine learning model that is trained on a desired property. One can then set the desired output, and uses gradients within the model to compute an input that yields the targeted output. This input maps directly to a atomic structure that yields a desired property. This strategy was the one taken in Refs. [34, 35, 36]. We also refer the reader to Ref. [31] for a thorough review of inverse design case studies.

This thesis is organized as follows. In Part II, we cover aspects of methodologies that were used to complete the results of this thesis. This includes electronic structure, the Thomas-Fermi model, and DFT in Chapter 2, density functional perturbation theory (DFPT), and the calculation of the Raman intensity from first principles in Chapter 3, fundamentals of quantum Monte Carlo (QMC) in Chapter 4, ML techniques relating to deep neural networks (DNNs) in Chapter 5, and electron transport with tight binding in Chapter 6. In Part III, we show results from 6 articles that were written for this thesis. The first 5 articles focus on the acceleration of the computation of nanoscale devices. These works include using various forms of DNNs to predict: total energies computed with KS-DFT Chapter 7, various energies and electron densities computed with KS-DFT in Chapter 8, kinetic energy densities, functional derivatives of kinetic energies, and electron densities computed with KS-DFT in Chapter 9, polarization, the infinite dielectric tensor, Born-effective charges, longitudinal optical transverse optical (LO-TO) splitting, and Raman spectra computed with DFPT in Chapter 10, and using VDNNs to predict energy densities calculated with diffusion Monte Carlo for graphene systems in Chapter 11. The final article, shown in Chapter 12, focuses on the design of a nanoscale device, where evolutionary strategies paired with ANNs were

used to rapidly explore a design space. Lastly, in Part IV we summarize the thesis.

In closing, we outline the major contributions of this thesis.

1. We are the first to apply deep CNNs to atomistic systems. This includes devising a real space representation of atomistic systems and using CNNs to make accurate predictions of DFT calculations.

2. We are the first to use deep CNNs (with built-in extensivity) as DFT energy functionals and the first to show that an external potential can be mapped to DFT energies using deep learning. We are the first to use deep convolutional inverse graphics networks to map external potentials to their corresponding electron densities. In addition, we show for the first time that cooperative learning can be done to predict all energy contributions simultaneously.

3. We invent voxel DNNs (VDNNs) and show for the first time that kinetic energy densities can be predicted and used to predict kinetic energies after integration. This allows for the first electron density calculated via direct minimization with a VDNN.

4. We invent a deep learning framework that combines deep CNNs with recent advancements in machine learning for quantum chemistry. Using this methodology we are the first to compute derivatives of response quantities for condensed matter systems.

5. Using VDNNs we are able to show, for the first time, that one can predict energies computed with diffusion Monte Carlo.

6. We are the first to use a neural network as a superoperator. In doing so, we are the first to use an evolutionary algorithm to design a graphene-based device which can physically separate valley currents.

Part II

Methodologies

CHAPTER 2

Ground State Properties of a Nanoscale System

In this chapter, we discuss the basic electronic structure needed to compute ground state properties of an atomistic system. To do so, we have used Ref. [8]. The theory presented in this section is used in Chapter 7, Chapter 8, Chapter 9, Chapter 10, and Chapter 11.

2.1 The Exact Hamiltonian

Understanding the ground state of a system is critical to understanding the behaviour of a material. The starting point is the Hamiltonian. Ignoring relativistic effects[1], the Hamiltonian of N_e interacting electrons and N_i interacting nuclei is

$$\begin{aligned} H &= -\frac{\hbar^2}{2m_e}\sum_{i=1}^{N_\mathrm{e}}\nabla_i^2 - \frac{\hbar^2}{2}\sum_{i=1}^{N_\mathrm{i}}\frac{\nabla_i^2}{m_i} \\ &\quad -\frac{e^2}{4\pi\epsilon_0}\sum_{i=1}^{N_\mathrm{e}}\sum_{j=1}^{N_\mathrm{i}}\frac{Z_j}{|\boldsymbol{r}_i-\boldsymbol{R}_j|} \\ &\quad +\frac{e^2}{8\pi\epsilon_0}\sum_{i=1}^{N_\mathrm{e}}\sum_{j\neq i}^{N_\mathrm{e}}\frac{1}{|\boldsymbol{r}_i-\boldsymbol{r}_j|} + \frac{e^2}{8\pi\epsilon_0}\sum_{i=1}^{N_\mathrm{i}}\sum_{j\neq i}^{N_\mathrm{i}}\frac{Z_iZ_j}{|\boldsymbol{R}_i-\boldsymbol{R}_j|}. \end{aligned} \tag{2.1}$$

The first line is the kinetic energy of the electrons and nuclei, respectively. The second is the Coulomb interaction between the electrons and nuclei. The last line is the Coulomb interactions between all electrons followed by the Coulomb interactions between all nuclei. The first simplification of this Hamiltonian which can be made is the separation between the electronic and ionic degrees of freedom. This is referred to as the Born-Oppenheimer

[1] This is a good approximation when studying nuclei that have low atomic mass.

approximation, or the adiabatic approximation. This allows one to describe the nuclei as classical point particles rather than quantum particles. This approximation removes the kinetic energy operator for the nuclei and implies that the Coulomb interactions between the nuclei is constant. The remaining three terms make up what is called the electronic Hamiltonian

$$H_{\mathrm{e}} = -\frac{\hbar^2}{2m_e}\sum_{i=1}^{N_{\mathrm{e}}}\nabla_i^2 - \frac{e^2}{4\pi\epsilon_0}\sum_{i=1}^{N_{\mathrm{e}}}\sum_{j=1}^{N_{\mathrm{i}}}\frac{Z_j}{|\boldsymbol{r}_i - \boldsymbol{R}_j|} + \frac{e^2}{8\pi\epsilon_0}\sum_{i=1}^{N_{\mathrm{e}}}\sum_{j\neq i}^{N_{\mathrm{e}}}\frac{1}{|\boldsymbol{r}_i - \boldsymbol{r}_j|}. \tag{2.2}$$

Next, we adopt atomic units. To do so, we perform variable changes for each coordinate by writing $\boldsymbol{u} = a\boldsymbol{r}$, $\boldsymbol{U} = a\boldsymbol{R}$ and rewrite the time-independant Schrödinger equation

$$\begin{aligned}
H_{\mathrm{e}}\Psi &= \epsilon\Psi \\
-\frac{\hbar^2}{2m_e a^2}\sum_{i=1}^{N_{\mathrm{e}}}\nabla_i^2\Psi - \frac{e^2}{4\pi\epsilon_0 a}\sum_{i=1}^{N_{\mathrm{e}}}\sum_{j=1}^{N_{\mathrm{i}}}\frac{Z_j}{|\boldsymbol{u}_i - \boldsymbol{U}_j|}\Psi + \frac{e^2}{8\pi\epsilon_0 a}\sum_{i=1}^{N_{\mathrm{e}}}\sum_{j\neq i}^{N_{\mathrm{e}}}\frac{1}{|\boldsymbol{u}_i - \boldsymbol{u}_j|}\Psi &= \epsilon\Psi \\
-\frac{1}{2}\sum_{i=1}^{N_{\mathrm{e}}}\nabla_i^2\Psi - \frac{e^2 m_e a}{4\pi\epsilon_0\hbar^2}\sum_{i=1}^{N_{\mathrm{e}}}\sum_{j=1}^{N_{\mathrm{i}}}\frac{Z_j}{|\boldsymbol{u}_i - \boldsymbol{U}_j|}\Psi + \frac{e^2 m_e a}{8\pi\epsilon_0\hbar^2}\sum_{i=1}^{N_{\mathrm{e}}}\sum_{j\neq i}^{N_{\mathrm{e}}}\frac{1}{|\boldsymbol{u}_i - \boldsymbol{u}_j|}\Psi &= \frac{m_e a^2}{\hbar^2}\epsilon\Psi.
\end{aligned} \tag{2.3}$$

Now defining $a = \frac{4\pi\epsilon_0\hbar^2}{e^2 m_e}$, we obtain

$$-\frac{1}{2}\sum_{i=1}^{N_{\mathrm{e}}}\nabla_i^2\Psi - \sum_{i=1}^{N_{\mathrm{e}}}\sum_{j=1}^{N_{\mathrm{i}}}\frac{Z_j}{|\boldsymbol{u}_i - \boldsymbol{U}_j|}\Psi + \frac{1}{2}\sum_{i=1}^{N_{\mathrm{e}}}\sum_{j\neq i}^{N_{\mathrm{e}}}\frac{1}{|\boldsymbol{u}_i - \boldsymbol{u}_j|}\Psi = \frac{8\pi^2\epsilon_0^2\hbar^2}{me^4}\epsilon\Psi. \tag{2.4}$$

The inverse of the collection of variables next to the eigenvalue ϵ is defined to be 1 Hartree or 1 Ha ($\approx$ 27.2 eV). To give some context, the ionization energy of H_2 is $\approx$ 15 eV [37]. Converting back to $\boldsymbol{r}$ and $\boldsymbol{R}$, our final electronic Hamiltonian in atomic units we must tackle is

$$H_{\mathrm{e}} = -\frac{1}{2}\sum_{i=1}^{N_{\mathrm{e}}}\nabla_i^2 - \sum_{i=1}^{N_{\mathrm{e}}}\sum_{j=1}^{N_{\mathrm{i}}}\frac{Z_j}{|\boldsymbol{r}_i - \boldsymbol{R}_j|} + \frac{1}{2}\sum_{i=1}^{N_{\mathrm{e}}}\sum_{j\neq i}^{N_{\mathrm{e}}}\frac{1}{|\boldsymbol{r}_i - \boldsymbol{r}_j|}. \tag{2.5}$$

Solving the time-independent Schrödinger equation, means solving for a set of wavefunctions (ignoring spin) and energies that satisfy the eigenvalue equation

$$H_{\mathrm{e}}\Psi_n(\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_{N_{\mathrm{e}}}) = \epsilon_n\Psi_n(\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_{N_{\mathrm{e}}}). \tag{2.6}$$

Writing the wavefunction Ψ_n in this way allows us to see the complexity of the problem. Our wavefunction is a function of 3^{N_e} variables. This exponential scaling inhibits directly solving this equation. If we consider discretizing each dimension with a grid spacing allowing for 10 values, then for one electron we need to store 1000 numbers, a million for two electrons, and a billion for three electrons. This equation quickly becomes infeasible to solve directly. This exponential scaling lead to other ways of solving a multi-electron system. This includes wavefunction and electron density based methods that exclude explicit electron correlation, wavefunction based methods that include electron correlation, and quantum Monte Carlo.

2.2 Density Functional Theory

2.2.1 The Thomas-Fermi Model

The exponential scaling of the multi-electron wavefunction was quickly realized early in the 20th century. Thomas and Fermi were the first to consider the electron density as the basic variable rather than the wavefunction [38, 39]. Instead of writing an expression to solve for the ground state wavefunction, they wrote an expression that depends on the electron density, ρ:

$$E[\rho] = T[\rho] + U_{\text{ee}}[\rho] + U_{\text{ei}}[\rho]. \tag{2.7}$$

This is a total energy expression that is a functional of the electron density. We now derive each term in Section 2.2.1. For the kinetic energy functional T, Thomas and Fermi both considered a non-interacting, free electron gas. For a single electron in free space, the Schrödinger equation is

$$-\frac{1}{2}\nabla^2\psi(\boldsymbol{r}) = \epsilon\psi_n(\boldsymbol{r}). \tag{2.8}$$

The solutions are plane waves

$$\psi_{\boldsymbol{k}}(\boldsymbol{r}) = \frac{1}{\sqrt{\Omega}}e^{i\boldsymbol{k}\cdot\boldsymbol{r}} \tag{2.9}$$

with $k^2 = 2\epsilon$, and Ω is the volume that contains the electron. With periodic boundary conditions, $\psi(\boldsymbol{r} + \boldsymbol{a}_i) = \psi(\boldsymbol{r})$ where $i = 1, 2, 3$ represents the unit cell vector i we find that $\boldsymbol{k} = 2\pi n/\boldsymbol{a}_i$, where $n \in \mathbb{N}$ and $\boldsymbol{a}_i$ a lattice constant in the direction i. With N_e electrons in our system, in the ground state we find that each state labelled by $\boldsymbol{k}$ will have 2 electrons (due to spin) up to the Fermi wavevector $\boldsymbol{k}_\mathrm{F}$. In reciprocal space, or $\boldsymbol{k}$ space, these filled states form a sphere with radius $\boldsymbol{k}_\mathrm{F}$. The total number of particles allows us to find a relationship between the density and the Fermi wavevector.

$$\begin{aligned} N &= 2\sum_{\boldsymbol{k}\leq\boldsymbol{k}_\mathrm{F}} \\ &= \frac{2\Omega}{(2\pi)^3}\int d\boldsymbol{k} && (2.10) \\ &= \frac{2\Omega}{(2\pi)^3}\int_0^{k_\mathrm{F}} k^2 dk \int_0^{2\pi} d\theta \int_0^{\pi} \sin\phi d\phi && (2.11) \\ N &= \frac{\Omega k_\mathrm{F}^3}{3\pi^2} && (2.12) \end{aligned}$$

which means than the electron density can be written as

$$\rho = \frac{k_\mathrm{F}^3}{3\pi^2}. \tag{2.13}$$

In a similar fashion, we can find the kinetic energy:

$$\begin{aligned} T &= 2\sum_{\boldsymbol{k}\leq\boldsymbol{k}_\mathrm{F}} \frac{k^2}{2} \\ &= \frac{\Omega}{(2\pi)^3}\int d\boldsymbol{k}\, k^2 && (2.14) \\ &= \frac{\Omega}{(2\pi)^3}\int_0^{k_\mathrm{F}} k^4 dk \int_0^{2\pi} d\theta \int_0^{\pi} \sin\phi d\phi && (2.15) \\ T &= \frac{\Omega k_\mathrm{F}^5}{10\pi^2}, && (2.16) \end{aligned}$$

or the kinetic energy density as

$$\mathcal{T} = \frac{k_\mathrm{F}^5}{10\pi^2}. \tag{2.17}$$

Inserting $k_{\mathrm{F}} = (3\pi^2\rho)^{1/3}$ into this expression we find

$$\begin{aligned} \mathcal{T}[\rho] &= \frac{(3\pi^2\rho)^{5/3}}{10\pi^2} \\ &= C_{\mathrm{F}}\rho^{5/3}. \end{aligned} \tag{2.18}$$

There is a simple interpretation of this expression: If the electron density increases, so does the kinetic energy density. Using this expression as an approximation to the kinetic energy density for all electron densities (i.e. non-uniform ones), we find

$$T[\rho(\boldsymbol{r})] = C_{\mathrm{F}} \int_{\Omega} d\boldsymbol{r}\ \rho^{5/3}(\boldsymbol{r}). \tag{2.19}$$

Now we consider the interactions between particles. For the electron-electron energy, we consider the electric potential of the electron density. We then consider how the electron density would interact with this electric potential. Mathematically, we write

$$U_{\mathrm{ee}}[\rho(\boldsymbol{r})] = \frac{1}{2} \int_{\Omega} d\boldsymbol{r} \int_{\Omega} d\boldsymbol{r}'\ \frac{\rho(\boldsymbol{r})\rho(\boldsymbol{r}')}{|\boldsymbol{r} - \boldsymbol{r}'|}. \tag{2.20}$$

The energy due to the electron-ion interactions can be defined after defining the electric potential for the set of static nuclei, $V_{\mathrm{ion}}(\boldsymbol{r})$, yielding

$$U_{\mathrm{ei}}[\rho(\boldsymbol{r})] = \int_{\Omega} d\boldsymbol{r}\ V_{\mathrm{ion}}(\boldsymbol{r})\rho(\boldsymbol{r}). \tag{2.21}$$

With these terms written, the goal is to find ρ such that the total energy is minimized. We differentiate the total energy expression with respect to ρ and set it to 0 (as done in any minimization problem). The only difference is that now, we have functional derivatives rather than partial derivatives. A functional derivative is found in the following way. Consider an arbitrary functional $F[\rho(\boldsymbol{r})]$. The functional derivative $\frac{\delta F}{\delta \rho}$ is found via

$$\int_{\Omega} d\boldsymbol{r}\ \frac{\delta F}{\delta \rho}\phi(\boldsymbol{r}) = \lim_{\epsilon \to 0} \frac{F[\rho + \epsilon\phi] - F[\rho]}{\epsilon} = \lim_{\epsilon \to 0} \frac{\partial}{\partial \epsilon} F[\rho + \epsilon\phi], \tag{2.22}$$

where ϵ is a small number, and $\phi(\boldsymbol{r})$ is an arbitrary function. For the Thomas-Fermi kinetic energy, we write

$$\begin{aligned}\int_\Omega d\boldsymbol{r}\ \frac{\delta T}{\delta\rho}\phi(\boldsymbol{r}) &= C_\mathrm{F}\lim_{\epsilon\to 0}\int_\Omega d\boldsymbol{r}\ \frac{\partial}{\partial\epsilon}(\rho(\boldsymbol{r})+\epsilon\phi(\boldsymbol{r}))^{5/3}\\ &= \frac{5C_\mathrm{F}}{3}\lim_{\epsilon\to 0}\int_\Omega d\boldsymbol{r}\ (\rho(\boldsymbol{r})+\epsilon\phi(\boldsymbol{r}))^{2/3}\phi(\boldsymbol{r})\\ &= \frac{5C_\mathrm{F}}{3}\int_\Omega d\boldsymbol{r}\ \rho(\boldsymbol{r})^{2/3}\phi(\boldsymbol{r}).\end{aligned} \tag{2.23}$$

Therefore, the functional derivative of the kinetic energy with respect to the density is

$$\frac{\delta T}{\delta\rho} = \frac{5C_\mathrm{F}}{3}\rho(\boldsymbol{r})^{2/3}. \tag{2.24}$$

Similarly, for the electron-electron energy we find

$$\begin{aligned}\int_\Omega d\boldsymbol{r}\ \frac{\delta U_\mathrm{ee}}{\delta\rho}\phi(\boldsymbol{r}) &= \frac{1}{2}\lim_{\epsilon\to 0}\int_\Omega d\boldsymbol{r}\int_\Omega d\boldsymbol{r}'\ \frac{\partial}{\partial\epsilon}\frac{\rho(\boldsymbol{r})\rho(\boldsymbol{r}')+\epsilon\rho(\boldsymbol{r})\phi(\boldsymbol{r}')+\epsilon\rho(\boldsymbol{r}')\phi(\boldsymbol{r})+\epsilon^2\phi(\boldsymbol{r})\phi(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|}\\ &= \frac{1}{2}\int_\Omega d\boldsymbol{r}\int_\Omega d\boldsymbol{r}'\ \frac{\rho(\boldsymbol{r})\phi(\boldsymbol{r}')+\rho(\boldsymbol{r}')\phi(\boldsymbol{r})}{|\boldsymbol{r}-\boldsymbol{r}'|}\\ &= \int_\Omega d\boldsymbol{r}\int_\Omega d\boldsymbol{r}'\ \frac{\rho(\boldsymbol{r}')\phi(\boldsymbol{r})}{|\boldsymbol{r}-\boldsymbol{r}'|}.\end{aligned} \tag{2.25}$$

The last step can be done because interchanging $\boldsymbol{r}$ and $\boldsymbol{r}'$ has no effect. Thus, the functional derivative of the electron-electron with respect to the density yields

$$\frac{\delta U_\mathrm{ee}}{\delta\rho} = \int_\Omega d\boldsymbol{r}'\ \frac{\rho(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|} \equiv V_\mathrm{Hartree}(\boldsymbol{r}), \tag{2.26}$$

which is also referred to as the Hartree potential. Lastly, we have the electron-ion energy. Proceeding as before, we find

$$\begin{aligned}\int_\Omega d\boldsymbol{r}\ \frac{\delta U_\mathrm{ei}}{\delta\rho}\phi(\boldsymbol{r}) &= \lim_{\epsilon\to 0}\int_\Omega d\boldsymbol{r}\ \frac{\partial}{\partial\epsilon}V_\mathrm{ion}(\boldsymbol{r})[\rho(\boldsymbol{r})+\epsilon\phi(\boldsymbol{r})]\\ &= \int_\Omega d\boldsymbol{r}\ V_\mathrm{ion}(\boldsymbol{r})\phi(\boldsymbol{r}),\end{aligned} \tag{2.27}$$

which means that

$$\frac{\delta U_\mathrm{ei}}{\delta\rho} = V_\mathrm{ion}(\boldsymbol{r}). \tag{2.28}$$

With the functional derivatives defined, we can start the optimization process. We first define the Lagrangian

$$\mathcal{L} = E[\rho] - \mu \left(\int_{\Omega} d\boldsymbol{r}\ \rho(\boldsymbol{r}) - N_{\mathrm{e}} \right), \tag{2.29}$$

where μ is a Lagrange multiplier for the condition that the number of electrons is fixed. To find the minimum of $\mathcal{L}$, we take the functional derivative with respect to ρ and set it to 0. This yields

$$\frac{\delta T[\rho]}{\delta \rho} + V_{\mathrm{ion}}(\boldsymbol{r}) + V_{\mathrm{Hartree}}(\boldsymbol{r}) - \mu = 0. \tag{2.30}$$

To solve for ρ, one technique is direct minimization where one first guesses at a starting value of ρ, and updates ρ by following the curvature defined by Equation 2.30 (e.g. gradient descent). This is a self-consistent procedure where one defines a tolerance in the change of energy from step-to-step. Once the energy changes are less than the defined tolerance, the ground state electron density and energy has been found using the Thomas-Fermi model. This procedure is also done in an orbital-free (OF) DFT calculation. However, OF-DFT calculations typically include more sophisticated approximations to the kinetic energy and exchange-correlation functionals that depend on the electron density.

The Thomas-Fermi model simplified the many-body problem by simply considering the electron density as the basic variable rather than the many-body wavefunction. This allowed for a computationally feasible, straightforward protocol to calculate an electron density by minimizing the Thomas-Fermi energy functional. However, its applicability is limited due to the lack of quantum interactions, and the fact that the kinetic energy is meant for non-interacting, free electrons. In addition to these limitations, there were still questions that had to be answered. Namely, does direct minimization of the "true" energy functional yield the exact ground state density? Is it possible to find another electron density in another local minimum during the optimization process?

2.2.2 Hohenberg-Kohn Theorems

Hohenberg and Kohn [40] answered the above questions with two proofs that led to two theorems:

1. The total energy of interacting electrons in some external potential V_{ext} is a unique functional of the electron density.
2. The minimum of the energy for fully interacting electrons has a lower bound with the true ground state electron density.

The fully interacting energy functional of the electron density can be written as

$$E^{\text{int}}[\rho] = F[\rho] + \int_{\Omega} d\boldsymbol{r}\, V_{\text{ext}}(\boldsymbol{r})\rho(\boldsymbol{r}), \tag{2.31}$$

where

$$F[\rho] = T_{\text{int}}[\rho] + U_{\text{ee}}^{\text{int}}[\rho] \tag{2.32}$$

is a universal functional of the density. It is universal because the many-body kinetic energy and Coulombic interactions between electrons are always the same for a system of N_{e} electrons. However, for a many-body system, F is unknown and must be approximated. Hohenberg and Kohn [40] extracted the classical electron-electron (Hartree) energy from F, and wrote

$$F[\rho] = \int_{\Omega} d\boldsymbol{r}\, V_{\text{Hartree}}(\boldsymbol{r})\rho(\boldsymbol{r}) + G[\rho], \tag{2.33}$$

where G is also a universal functional. Afterwards, they examined the form of G for an almost constant and a slowly varying electron density. Although the formalities of DFT were addressed, better quantum mechanical kinetic and electron-electron energies were still missing. This was addressed by Kohn and Sham [41].

2.2.3 Kohn-Sham Theory

One year after Hohenberg and Kohn [40] introduced the theoretical framework for DFT, Kohn and Sham [41] devised a practical implementation of DFT that included a quantum mechanical form for the kinetic energy. In addition, they introduced an approximation to the exchange-correlation effects of interacting electrons. This was accomplished by comparing the electron density of a many-body system to the electron density from single-particle orbitals. On one hand, if one had access to the many-body wavefunction, one can write the electron density as

$$\rho(\boldsymbol{r}) = N_{\mathrm{e}} \int_{\Omega} d\boldsymbol{r}_2 \int_{\Omega} d\boldsymbol{r}_3 \ \ldots \int_{\Omega} d\boldsymbol{r}_{N_{\mathrm{e}}} \, |\Psi(\boldsymbol{r}, \boldsymbol{r}_2, \boldsymbol{r}_3, \ldots, \boldsymbol{r}_{N_{\mathrm{e}}})|^2. \tag{2.34}$$

On the other hand, as shown graphically in Figure 2.1, one can also consider an electron density constructed from single particle (Kohn-Sham) orbitals

$$\rho(\boldsymbol{r}) = 2 \sum_{i=1}^{N_{\mathrm{e}}/2} |\psi_i(\boldsymbol{r})|^2. \tag{2.35}$$

Is it possible to create an auxiliary system such that these electron densities are the same? For the single-particle system, one can approximate the universal functional G from Equation 2.33 by writing

$$G[\rho] = T_{\text{non-int}}[\rho] + E_{\mathrm{xc}}[\rho] \tag{2.36}$$

where

$$T_{\text{non-int}}[\rho] = -\sum_{i=1}^{N_{\mathrm{e}}/2} \langle \psi_i | \nabla^2 | \psi_i \rangle \tag{2.37}$$

is the non-interacting kinetic energy and

$$E_{\mathrm{xc}}[\rho] = \int_{\Omega} d\boldsymbol{r} \, \epsilon_{\mathrm{xc}}(\rho(\boldsymbol{r}))\rho(\boldsymbol{r}) \tag{2.38}$$

is the exchange-correlation energy. What exactly is the physical significance of the exchange-correlation energy? Well, if we consider the true, fully interacting energy written in Equation 2.31, and assume that the energy from our approximation of the universal functional G

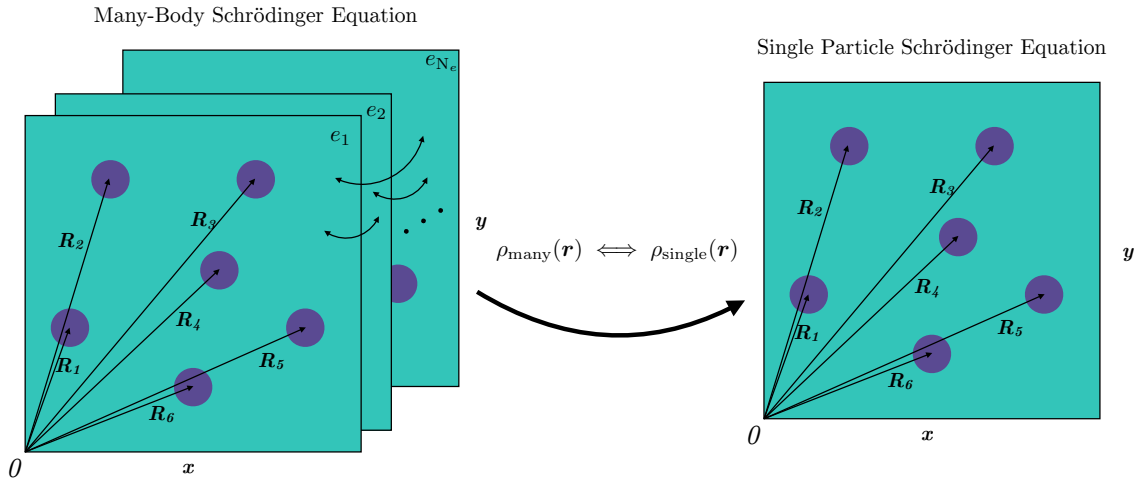


Figure 2.1: A 2-dimensional graphical representation of the approximation made in Kohn-Sham density functional theory. Purple circles represent nuclei, and the blue surfaces represent the electron coordinates. In the many-body Schrödinger picture, we have electron-electron interactions, leading to a many-body wavefunction that can be integrated over to yield the electron density. In the single particle Schrödinger picture, we have a single electron system that includes potentials such that the many-body electron density is reproduced. The exchange-correlation potential is introduced to approximate this reproduction.

yields the same value as the fully interacting energy, then

$$E_{\text{xc}}[\rho] = T_{\text{int}}[\rho] - T_{\text{non-int}}[\rho] + U_{\text{ee}}^{\text{int}}[\rho] - U_{\text{Hartree}}[\rho]. \tag{2.39}$$

Therefore, we see that E_{xc} is the difference between the true, fully interacting kinetic and electron-electron energies and non-interacting kinetic and classical electron-electron energies. It describes the remaining quantum nature of the electrons, which includes dynamic correlation of the electron coordinates and the antisymmetric properties of the many-body wavefunction with the exchange of two electron coordinates. If known, one is left solving for single-particle orbitals which yield the exact ground state electron density and energy. However, this expression is not known and must be approximated. Kohn and Sham proposed the first exchange-correlation functional assuming a free electron gas [41]. This is called the local density approximation (LDA). Within the LDA, the exchange-

correlation energy is split into separate exchange and correlation parts

$$E_{\text{xc}}[\rho] = \int_{\Omega} d\boldsymbol{r}\ (\epsilon_{\text{x}}(\rho(\boldsymbol{r})) + \epsilon_{\text{c}}(\rho(\boldsymbol{r})))\rho(\boldsymbol{r}). \tag{2.40}$$

For the exchange part, Kohn and Sham (and Slater [42]) considered the exchange operator from Hartree-Fock theory and wrote and equivalent potential that acts on the n^{th} orbital

$$V_{\text{x},n}(\boldsymbol{r}) = -\frac{1}{\psi_n^*(\boldsymbol{r})\psi_n(\boldsymbol{r})} \sum_{m=1}^{N_{\text{el}}/2} \int_{\Omega} d\boldsymbol{r}'\ \frac{\psi_n^*(\boldsymbol{r})\psi_m^*(\boldsymbol{r}')\psi_m(\boldsymbol{r})\psi_n(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|}. \tag{2.41}$$

For a non-interacting electron gas where orbitals are plane waves labelled by wavenumber $\boldsymbol{k}$ the result is

$$E_x[\rho] = -\frac{3(3\pi^2)^{1/3}}{2\pi} \int_{\Omega} d\boldsymbol{r}\ \rho(\boldsymbol{r})^{4/3}. \tag{2.42}$$

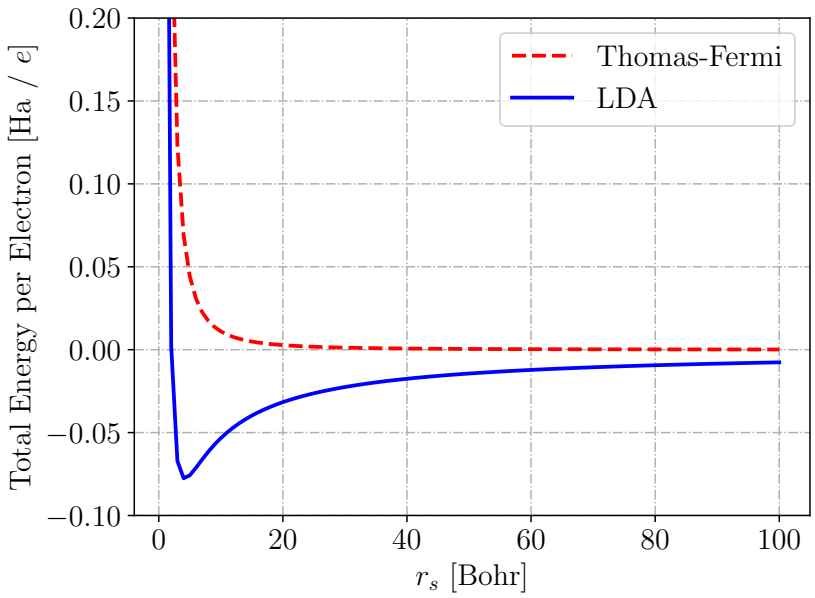


Figure 2.2: Total energy per electron of a free electron gas as a function of r_s. The Thomas-Fermi model is strictly positive. However, with the addition of electron correlation and exchange, the energy becomes negative with a minimum at $r_s \approx 5$ Bohr.

As the electron density increases, the exchange energy decreases. This can also be seen in Figure 2.2, where the addition of the exchange-correlation term causes the energy to become negative for $r_s > 2$. Low values of r_s correspond to high electron densities. As $r_s \to 1$, the kinetic energy becomes larger than exchange-correlation, and we see a positive energy. For

the correlation energy, calculations of interacting electron gases were performed at differing densities by Ceperley and Alder [43] with diffusion Monte Carlo. In this case, one can subtract the Hartree-Fock total energy from the QMC total energy to to obtain the correlation energy. These energies were then fit to analytic forms which are shown in Ref. [8]. Other functionals have also been developed to improve the LDA. Two popular approximations are Perdew-Burke-Erzerhof (PBE) [44], and Becke-3 paramater-Lee-Yang-Parr (B3LYP) [45]. PBE is classified under the generalized gradient approximation (GGA), and it utilizes both the electron density and gradients of the electron density. For this reason, GGA functionals are considered semi-local. Generally, the exchange-correlation energies for GGAs are written (ignoring spin) as [8]

$$E_{\mathrm{xc}}^{\mathrm{GGA}}[\rho] = \int_{\Omega} d\boldsymbol{r}\ \rho(\boldsymbol{r})(\epsilon_{\mathrm{x}}^{\mathrm{hom}}(\rho)F_{\mathrm{x}}[\rho, |\nabla\rho|, \ldots] + \epsilon_{\mathrm{c}}^{\mathrm{hom}}(\rho)F_{\mathrm{c}}[\rho, |\nabla\rho|, \ldots]) \tag{2.43}$$

where ϵ^{hom} is the exchange/correlation energy per particle for a non-interacting electron gas, as discussed above. The dots within F indicate that further derivatives could be used in the approximation. For the PBE functional, only single derivatives of the density are used and the forms of F are chosen such that certain physical conditions are satisfied. The B3LYP functional is classified as a hybrid functional because it combines exact exchange (Equation 2.41) for the Kohn-Sham orbitals with terms from both LDA and GGA functionals. The parameters in this functional are the coefficients of each term. B3LYP is more computationally demanding in comparison to LDA or GGA due to the inclusion of exact exchange, which must be numerically computed for a set of Kohn-Sham orbitals.

With our approximate total energy expression, we perform a calculation similar to Equation 2.29, but now with respect to a Kohn-Sham orbital ψ_i, rather than the density. This yields a single-particle Schrödinger equation (or Kohn-Sham equation) for the set of orbitals

$$\left[-\frac{1}{2}\nabla^2 + V_{\mathrm{ext}}(\boldsymbol{r}) + V_{\mathrm{Hartree}}(\boldsymbol{r}) + \frac{\delta E_{\mathrm{xc}}[\rho]}{\delta\rho}\right]\psi_i = \epsilon_i\psi_i. \tag{2.44}$$

The many-body problem has now been mapped onto a single-particle eigenvalue problem, where we calculate the lowest $N_{\mathrm{e}}/2$ eigenvalues and eigenvectors and fill them with

electrons. A self-consistent procedure, similar to Equation 2.30, must be done to find the approximated ground state energy and electron density.

2.2.4 Additional Details

In this section, we discuss other details that go into a KS-DFT calculation. First is the use of pseudopotentials. Rather than using the bare Coulomb potential for the nuclei, a new potential is introduced to reproduce the environment of a valence electron. Around an atom, core electrons are tightly bound and are not involved in bonding. We combine the potential due to these frozen core electrons and the potential of the nuclei to create what is called a pseudopotential. With pseudopotentials, we only calculate the orbitals for the valence electrons which reduces the number of orbitals that must be found when solving the eigenvalue problem. In addition, if we were to solve for the orbitals for core electrons, higher lying states having more nodes and are more oscillatory than lower lying ones. This becomes challenging to represent on a real space grid.

Second, is the use of a $\boldsymbol{k}$-point grid. In KS-DFT with periodic boundary conditions the orbitals are Bloch states such that

$$\psi_{n,\boldsymbol{k}}(\boldsymbol{r}) = u_{n,\boldsymbol{k}}(\boldsymbol{r})e^{i\boldsymbol{k}\cdot\boldsymbol{r}} \tag{2.45}$$

where $\boldsymbol{k} = 2\pi m/\boldsymbol{a}_i$ ($m = 1, 2, 3, \ldots$, $\boldsymbol{a}_i$ is a unit cell vector) and n represents the band index. The function $u_{n,\boldsymbol{k}}(\boldsymbol{r})$ is the periodic part of the orbital. Substituting this into Equation 2.44 yields an eigenvalue problem for u given a particular wavevector $\boldsymbol{k}$. When calculating the energy of a periodic system, we must consider energy per unit cell and how it changes as a function of the number of unit cells. The energy will converge as we increase the number of cells. This can also be achieved by considering the $\boldsymbol{k}$ points. Each $\boldsymbol{k}$ point corresponds to a particular unit cell, and instead of increasing the number of unit cells, we increase the number of $\boldsymbol{k}$-points. Again, we will see the convergence of the energy as we increase the number of $\boldsymbol{k}$-points. In a typical KS-DFT code, each eigenvalue equation, labelled by the wavenumber $\boldsymbol{k}$, are solved in parallel. Each $\boldsymbol{k}$-point can be mapped into the Brillouin zone

and quantities are averaged over the Brillouin zone. For example the electron density is written as

$$\rho(\boldsymbol{r}) = \frac{1}{N_{\boldsymbol{k}}} \sum_{n=1}^{N_{\mathrm{e}}/2} \sum_{m=1}^{N_{\boldsymbol{k}}} u^*_{n,\boldsymbol{k}_m}(\boldsymbol{r}) u_{n,\boldsymbol{k}_m}(\boldsymbol{r}). \tag{2.46}$$

Third, is the use of plane waves as a basis the periodic part of the orbital, $u(\boldsymbol{r})$. This is written as

$$u_{n,\boldsymbol{k}}(\boldsymbol{r}) = \sum_{\boldsymbol{G}}^{\boldsymbol{G}<|\boldsymbol{G}_{\mathrm{max}}|} u_{n,\boldsymbol{k}}(\boldsymbol{G}) e^{i\boldsymbol{G}\cdot\boldsymbol{r}}, \tag{2.47}$$

where $\boldsymbol{G}$ is also a wavevector that satisfies the same equation as $\boldsymbol{k}$. The sum only includes $\boldsymbol{G}$ vectors such that

$$E_{\mathrm{cut}} = \frac{|\boldsymbol{G}_{\mathrm{max}}|^2}{2} \tag{2.48}$$

where E_{cut} is a maximum kinetic energy cut-off that must be defined before performing a calculation. The larger the kinetic energy cut-off, the more basis functions are included in the calculation. Doing this reduces the size of the Hamiltonian matrix (compared to a real space calculation) that is being used in the eigenvalue problem, therefore reducing the computational cost. However, one must increase the energy cut-off such that the total energy has converged.

Chapter 3

Calculating Raman Spectra with *ab initio* Methods

The Raman spectra is the distribution of vibrational energies for a particular sample. In Raman spectroscopy one shines light with a fixed wavelength on a sample and there is momentum exchange between the incoming light and the phonons. This results in the outgoing light possibly having a different frequency compared to the incoming light. If a Raman-active phonon exists before light is emitted onto the sample, it is possible for an outgoing photon to have a higher frequency in comparison with the frequency of the incoming photon. The increase in frequency would be the exact frequency of the phonon that was annihilated in the process, which is known as an anti-Stokes scattering. If a Raman-active phonon does not exist, another possibility is the incoming photon excites the phonon, and the outgoing photon has a lower frequency in comparison to the incoming photon. The difference between the frequencies is exactly equal to the frequency of the phonon, and this process is known as Stokes scattering. In either case, the vibrational energy, or phonon energy can be extracted via the difference in energy between the incoming and outgoing light. These two processes can be seen in Figure 3.1. In this chapter, we cover a density functional theory framework that allows for the computation of response quantities needed for Raman spectra. This framework is called density functional perturbation theory (DFPT). For this chapter we follow Ref. [46].

3.1 Introduction to Density Functional Perturbation Theory

Let us first consider lattice dynamics. After invoking the Born-Oppenheimer approximation (or the adiabatic approximation), we can solve for the total electronic energy of the system.

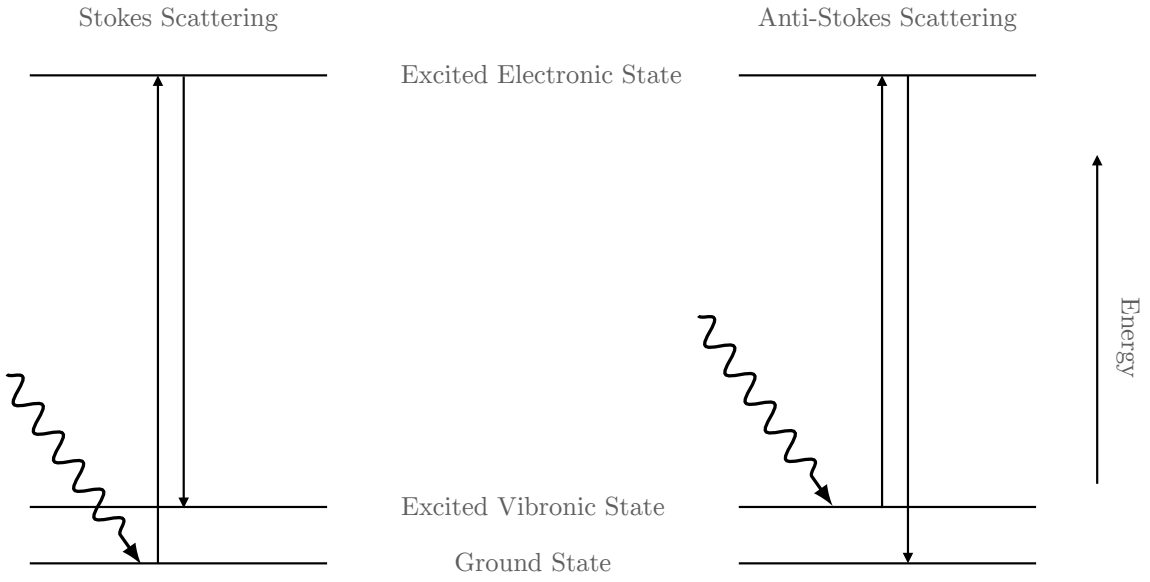


Figure 3.1: Energy level diagram for Stokes and Anti-Stokes scattering which can be seen in a Raman spectra. An incoming photon with a well defined frequency first excites the system to a higher lying electronic state. When the system relaxes to a lower lying state a photon is emitted. In the Stokes process the excitation creates a phonon and the frequency of the emitted light is less than the incoming light. In the Anti-Stokes process, a phonon is annihilated and the frequency of the emitted light is higher than the incoming light.

This energy surface has a dependence on the electronic coordinates and it also has a parametric dependence on the nuclear coordinates $E(\{\boldsymbol{r}\},\{\boldsymbol{R}\})$. Here, $\{\boldsymbol{r}\} = \boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_n$ is the set of electron coordinates and $\{\boldsymbol{R}\} \equiv \boldsymbol{R}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_N$ is the set of nuclear coordinates. In the equilibrium geometry, the force on the nuclei I in direction j is

$$F_{Ij} = -\frac{\partial E(\{\boldsymbol{r}\},\{\boldsymbol{R}\})}{\partial R_{Ij}} = 0. \tag{3.1}$$

How do we find such a derivative? We can use the Hellman-Feynman theorem which states that the derivative of the eigenvalues of a Hamiltonian H_λ, which depends on a parameter λ is

$$\frac{\partial E(\lambda)}{\partial \lambda} = \left\langle \Psi_0(\{\boldsymbol{r}\},\lambda) \left| \frac{\partial H(\{\boldsymbol{r}\},\lambda)}{\partial \lambda} \right| \Psi_0(\{\boldsymbol{r}\},\lambda) \right\rangle, \tag{3.2}$$

In the above equation, the parameter λ can be replaced by a nuclear coordinate, Ψ_0 is the ground state wavefunction and the integration is over the electronic coordinates only. The

Born-Oppenheimer Hamiltonian is

$$H_{\text{BO}}(\{\boldsymbol{r}\},\{\boldsymbol{R}\}) = T_{\text{e}}(\{\boldsymbol{r}\}) + V_{\text{ee}}(\{\boldsymbol{r}\}) + V_{\text{ei}}(\{\boldsymbol{r}\},\{\boldsymbol{R}\}) + V_{\text{ii}}(\{\boldsymbol{R}\}), \tag{3.3}$$

where T_{e} describes the kinetic energy of the electrons, V_{ee} is the interactions of electrons, V_{ei} describes the interactions between electrons and nuclei, and V_{ii} describes the interaction between nuclei. When we differentiate with respect to a nuclear coordinate $\boldsymbol{R}_I$ in direction j, we find that

$$\begin{aligned} F_{Ij} = -\frac{\partial E(\{\boldsymbol{R}\})}{\partial R_{Ij}} &= -\left\langle \Psi_0(\{\boldsymbol{r}\},\{\boldsymbol{R}\}) \left| \frac{\partial H_{\text{BO}}(\{\boldsymbol{r}\},\{\boldsymbol{R}\})}{\partial R_{Ij}} \right| \Psi_0(\{\boldsymbol{r}\},\{\boldsymbol{R}\}) \right\rangle \\ &= -\left\langle \Psi_0(\{\boldsymbol{r}\},\{\boldsymbol{R}\}) \left| \frac{\partial}{\partial R_{Ij}} [V_{\text{ei}}(\{\boldsymbol{r}\},\{\boldsymbol{R}\}) + V_{\text{ii}}(\{\boldsymbol{R}\})] \right| \Psi_0(\{\boldsymbol{r}\},\{\boldsymbol{R}\}) \right\rangle \\ &= -\int d\boldsymbol{r}\ \rho(\{\boldsymbol{r}\},\{\boldsymbol{R}\}) \frac{\partial V_{\text{ei}}(\{\boldsymbol{r}\},\{\boldsymbol{R}\})}{\partial R_{Ij}} - \frac{\partial V_{\text{ii}}(\{\boldsymbol{R}\})}{\partial R_{Ij}} \end{aligned} \tag{3.4}$$

where $\rho(\{\boldsymbol{r}\},\{\boldsymbol{R}\})$ is the ground state electron density. The Hessian can also be computed by differentiating the energy again with respect to some coordinate $\boldsymbol{R}_J$ in a direction k. This Hessian is used to solve for phonon energies and atomic displacement vectors. Proceeding we find

$$\begin{aligned} \frac{\partial F_{IJ,jk}}{\partial R_{Jk}} = -\frac{\partial^2 E(\{\boldsymbol{r}\},\{\boldsymbol{R}\})}{\partial R_{Ij}\partial R_{Jk}} &= -\int d\boldsymbol{r}\ \frac{\partial \rho(\{\boldsymbol{r}\},\{\boldsymbol{R}\})}{\partial R_{Jk}} \frac{\partial V_{\text{ei}}(\{\boldsymbol{r}\},\{\boldsymbol{R}\})}{\partial R_{Ij}} \\ &\quad -\int d\boldsymbol{r}\ \rho(\{\boldsymbol{r}\},\{\boldsymbol{R}\}) \frac{\partial^2 V_{\text{ei}}(\{\boldsymbol{r}\},\{\boldsymbol{R}\})}{\partial R_{Ij}\partial R_{Jk}} - \frac{\partial^2 V_{\text{ii}}(\{\boldsymbol{R}\})}{\partial R_{Ij}\partial R_{Jk}}, \end{aligned} \tag{3.5}$$

where we find that we must differentiate the electron density with respect to one of the nucleic coordinates. This is exactly where perturbation theory comes into play. In perturbation theory, we begin by writing

$$E = E^{(0)} + \lambda E^{(1)} + \lambda^2 E^{(2)} + \dots \tag{3.6}$$

$$|\psi\rangle = |\psi\rangle^{(0)} + \lambda|\psi\rangle^{(1)} + \lambda^2|\psi\rangle^{(2)} + \dots \tag{3.7}$$

where λ is a small value that we expand the series around. To make it clear for the following discussion, we could also write this expansion as

$$E = E + \lambda\frac{\partial E}{\partial \lambda} + \lambda^2\frac{\partial^2 E}{\partial \lambda^2} + \dots \tag{3.8}$$

$$|\psi\rangle = |\psi\rangle + \lambda\frac{\partial |\psi\rangle}{\partial \lambda} + \lambda^2\frac{\partial^2 |\psi\rangle}{\partial \lambda^2} + \dots \tag{3.9}$$

We can see here that the first order corrections of some variable are equivalent to finding derivatives of that variable with respect to the perturbation.

In the case of phonons, we say that the displacement of the nuclei in the system is small from their equilibrium positions. The displacement from the equilibrium geometry is labelled $\boldsymbol{u}$. The energy expression and the electron density can be parameterized by $\boldsymbol{u} \equiv \boldsymbol{R} - \boldsymbol{R}_0$ ($\boldsymbol{R}_0$ is the equilibrium geometry) and expanded about this small displacement $\boldsymbol{u}$ in direction i as

$$E(\boldsymbol{u}) = E(u_i = 0) - u_i\frac{\partial E(\boldsymbol{u})}{\partial u_i}\bigg|_{u_i=0} + \frac{u_i^2}{2}\frac{\partial^2 E(\boldsymbol{u})}{\partial u_i^2}\bigg|_{u_i=0} + \dots \tag{3.10}$$

$$n(\boldsymbol{u}) = n(u_i = 0) - u_i\frac{\partial n(\boldsymbol{u})}{\partial u_i}\bigg|_{u_i=0} + \frac{u_i^2}{2}\frac{\partial^2 n(\boldsymbol{u})}{\partial u_i^2}\bigg|_{u_i=0} + \dots \tag{3.11}$$

or

$$E(\boldsymbol{R}) = E(\boldsymbol{R}_0) - u_i\frac{\partial E(\boldsymbol{R})}{\partial R_i}\bigg|_{\boldsymbol{R}=\boldsymbol{R}_0} + \frac{u_i^2}{2}\frac{\partial^2 E(\boldsymbol{R})}{\partial R_i^2}\bigg|_{\boldsymbol{R}=\boldsymbol{R}_0} + \dots \tag{3.12}$$

$$n(\boldsymbol{R}) = n(\boldsymbol{R}_0) - u_i\frac{\partial n(\boldsymbol{R})}{\partial R_i}\bigg|_{\boldsymbol{R}=\boldsymbol{R}_0} + \frac{u_i^2}{2}\frac{\partial^2 n(\boldsymbol{R})}{\partial R_i^2}\bigg|_{\boldsymbol{R}=\boldsymbol{R}_0} + \dots \tag{3.13}$$

We can therefore see that finding the derivative of the electron density is equivalent to finding the first order expansion of the electron density. The goal now is to find a way to calculate the first order correction to the density. In KS-DFT, we write the density as a summation over the occupied Kohn-Sham orbitals in the Brillouin zone

$$n(\boldsymbol{r}, \{\boldsymbol{R}\}) = \frac{2}{N_{\boldsymbol{k}}}\sum_{\boldsymbol{k}}^{\text{BZ}}\sum_{n=1}^{\text{occ}}\psi^*_{n,\boldsymbol{k}}(\boldsymbol{r}, \{\boldsymbol{R}\})\psi_{n,\boldsymbol{k}}(\boldsymbol{r}, \{\boldsymbol{R}\}). \tag{3.14}$$

One way to calculate the first order correction to the charge density is to expand the Kohn-

Sham orbitals to first order and linearize. Proceeding yields

$$\begin{aligned} n(\boldsymbol{r},\{\boldsymbol{R}\}) &= \frac{2}{N_{\boldsymbol{k}}}\sum_{\boldsymbol{k}}^{\text{BZ}}\sum_{n=1}^{\text{occ}}[\psi^{(0)*}_{n,\boldsymbol{k}}(\boldsymbol{r},\{\boldsymbol{R}\})+\psi^{(1)*}_{n,\boldsymbol{k}}(\boldsymbol{r},\{\boldsymbol{R}\})][\psi^{(0)}_{n,\boldsymbol{k}}(\boldsymbol{r},\{\boldsymbol{R}\})+\psi^{(1)}_{n,\boldsymbol{k}}(\boldsymbol{r},\{\boldsymbol{R}\})) \\ &= \frac{2}{N_{\boldsymbol{k}}}\sum_{\boldsymbol{k}}^{\text{BZ}}\sum_{n=1}^{\text{occ}}\psi^{(0)*}_{n,\boldsymbol{k}}(\boldsymbol{r},\{\boldsymbol{R}\})\psi^{(0)}_{n,\boldsymbol{k}}(\boldsymbol{r},\{\boldsymbol{R}\}) \\ &\quad+\frac{2}{N_{\boldsymbol{k}}}\sum_{\boldsymbol{k}}^{\text{BZ}}\sum_{n=1}^{\text{occ}}\psi^{(0)*}_{n,\boldsymbol{k}}(\boldsymbol{r},\{\boldsymbol{R}\})\psi^{(1)}_{n,\boldsymbol{k}}(\boldsymbol{r},\{\boldsymbol{R}\})+\psi^{(0)}_{n,\boldsymbol{k}}(\boldsymbol{r},\{\boldsymbol{R}\})\psi^{(1)*}_{n,\boldsymbol{k}}(\boldsymbol{r},\{\boldsymbol{R}\}) \\ &= n^{(0)}(\boldsymbol{r},\{\boldsymbol{R}\})+n^{(1)}(\boldsymbol{r},\{\boldsymbol{R}\}) \end{aligned} \tag{3.15}$$

Now we must find a way to compute $\psi_i^{(1)}$. Without a perturbation, the Kohn-Sham Hamiltonian, as written in Section 2.2.3 is

$$H^{(0)}_{\text{KS}} \equiv -\frac{1}{2}\nabla^2 + V_{\text{ext}}(\boldsymbol{r}) + \int d\boldsymbol{r}'\,\frac{n(\boldsymbol{r}',\{\boldsymbol{R}\})}{|\boldsymbol{r}-\boldsymbol{r}'|} + V_{\text{xc}}(n(\boldsymbol{r},\{\boldsymbol{R}\})). \tag{3.16}$$

The final term is the exchange-correlation potential which is defined as the functional derivative of the exchange-correlation energy E_{xc} with respect to the electron density:

$$V_{\text{xc}}(n(\boldsymbol{r},\{\boldsymbol{R}\})) = \frac{\delta E_{\text{xc}}}{\delta n(\boldsymbol{r},\{\boldsymbol{R}\})}. \tag{3.17}$$

With a perturbation turned on the external potential is now $V_{\text{ext}}(\boldsymbol{r}) = V^{(0)}_{\text{ext}}(\boldsymbol{r}) + V^{(1)}_{\text{ext}}(\boldsymbol{r})$ where $V^{(1)}_{\text{ext}}(\boldsymbol{r})$ is the perturbation and the electron density is $n(\boldsymbol{r},\{\boldsymbol{R}\}) = n^{(0)}(\boldsymbol{r},\{\boldsymbol{R}\}) + n^{(1)}(\boldsymbol{r},\{\boldsymbol{R}\})$. The Kohn-Sham Hamiltonian then becomes

$$\begin{aligned} H_{\text{KS}} &= -\frac{1}{2}\nabla^2 + V^{(0)}_{\text{ext}}(\boldsymbol{r}) + V^{(1)}_{\text{ext}}(\boldsymbol{r}) + \int d\boldsymbol{r}'\,\frac{n^{(0)}(\boldsymbol{r},\{\boldsymbol{R}\})+n^{(1)}(\boldsymbol{r},\{\boldsymbol{R}\})}{|\boldsymbol{r}-\boldsymbol{r}'|} \\ &\quad+V_{\text{xc}}(n^{(0)}(\boldsymbol{r},\{\boldsymbol{R}\})+n^{(1)}(\boldsymbol{r},\{\boldsymbol{R}\})) \\ &= -\frac{1}{2}\nabla^2 + V^{(0)}_{\text{ext}}(\boldsymbol{r}) + V^{(1)}_{\text{ext}}(\boldsymbol{r}) + \int d\boldsymbol{r}'\,\frac{n^{(0)}(\boldsymbol{r},\{\boldsymbol{R}\})+n^{(1)}(\boldsymbol{r},\{\boldsymbol{R}\})}{|\boldsymbol{r}-\boldsymbol{r}'|} \\ &\quad+V_{\text{xc}}(n^{(0)}(\boldsymbol{r},\{\boldsymbol{R}\}))+n^{(1)}(\boldsymbol{r},\{\boldsymbol{R}\})\,\frac{\delta V_{\text{xc}}(n(\boldsymbol{r},\{\boldsymbol{R}\}))}{\delta n(\boldsymbol{r},\{\boldsymbol{R}\})}\bigg|_{n=n^{(0)}} \\ &= H^{(0)}_{\text{KS}} + H^{(1)}_{\text{KS}} \end{aligned} \tag{3.18}$$

where the perturbative Hamiltonian is

$$H_{\text{KS}}^{(1)} = V_{\text{ext}}^{(1)}(\boldsymbol{r}) + \int d\boldsymbol{r}\, \frac{n^{(1)}(\boldsymbol{r},\{\boldsymbol{R}\})}{|\boldsymbol{r}-\boldsymbol{r}'|} + n^{(1)}(\boldsymbol{r},\{\boldsymbol{R}\})\, \frac{\delta V_{\text{xc}}(n(\boldsymbol{r},\{\boldsymbol{R}\}))}{\delta n(\boldsymbol{r},\{\boldsymbol{R}\})}\bigg|_{n=n^{(0)}}. \tag{3.19}$$

In the perturbative Hamiltonian, the additional functional derivative comes from Taylor expanding about $n^{(1)} = 0$ to first order. Using standard perturbation theory we find that

$$\begin{aligned} H_{KS}^{(0)}\psi^{(1)} + H_{KS}^{(1)}\psi^{(0)} &= \epsilon^{(1)}\psi^{(0)} + \epsilon^{(0)}\psi^{(1)} \\ (H_{KS}^{(0)} - \epsilon^{(0)})\psi^{(1)} &= -(H_{KS}^{(1)} - \epsilon^{(1)})\psi^{(0)} \end{aligned} \tag{3.20}$$

which is called the Sternheimer equation. We solve a similar self-consistent procedure for the first order correction to the Kohn-Sham orbitals.

3.2 The Computation of the Electronic Susceptibility

In this section we describe one methodology to compute the electronic susceptibility tensor of a system with periodic boundary conditions. A quantity needed to compute the Raman tensor is a third order derivative of the energy which is written as

$$\alpha_{jk}^{i} = \frac{\partial^3 E}{\partial u_i \partial \mathcal{E}_j \partial \mathcal{E}_k} \tag{3.21}$$

where u_i is the displacement of the atom in direction i and $\mathcal{E}_j$ is an electric field in direction j. This expression can alternatively be written as

$$\alpha_{jk}^{i} = \frac{\partial \chi_{jk}}{\partial u_i} = \Omega \frac{\partial}{\partial u_i}\frac{\partial P_k}{\partial \mathcal{E}_j} \tag{3.22}$$

where $\boldsymbol{\chi}$ is the electronic susceptibility and $\boldsymbol{P}$ is the polarization. To obtain this quantity, we first consider the polarization. In the modern theory of polarization [47], one writes the electronic contribution to the polarization as

$$\boldsymbol{P} = \frac{1}{(2\pi)^3}\text{Im}\sum_{n=1}^{N_{\text{el}}/2}\int_{\text{BZ}} d\boldsymbol{k}\, \langle\psi_{n\boldsymbol{k}}|\nabla_{\boldsymbol{k}}|\psi_{n\boldsymbol{k}}\rangle. \tag{3.23}$$

Let's consider one component of the polarization, P_i, and apply a derivative of the electric field in one direction, $\mathcal{E}_j$. We find

$$\frac{\partial P_i}{\partial \mathcal{E}_j} = \frac{1}{(2\pi)^3}\mathrm{Im}\sum_{n=1}^{N_{\mathrm{el}}/2}\int_{\mathrm{BZ}} d\boldsymbol{k}\left\langle \frac{\partial \psi_{n\boldsymbol{k}}}{\partial \mathcal{E}_j}\middle|\frac{\partial}{\partial k_i}\middle|\psi_{n\boldsymbol{k}}\right\rangle + \left\langle \psi_{n\boldsymbol{k}}\middle|\frac{\partial^2}{\partial \mathcal{E}_j \partial k_i}\middle|\psi_{n\boldsymbol{k}}\right\rangle. \tag{3.24}$$

As shown in Section 3.1, the derivatives in the above expression can be considered corrections to the Kohn-Sham orbitals when a perturbation is introduced. We can therefore write

$$\frac{\partial P_i}{\partial \mathcal{E}_j} = \frac{1}{(2\pi)^3}\mathrm{Im}\sum_{n=1}^{N_{\mathrm{el}}/2}\int_{\mathrm{BZ}} d\boldsymbol{k}\,\langle \psi_{n\boldsymbol{k}}^{\mathcal{E}_j}|\psi_{n\boldsymbol{k}}^{k_i}\rangle + \langle \psi_{n\boldsymbol{k}}|\psi_{n\boldsymbol{k}}^{\mathcal{E}_j,k_i}\rangle. \tag{3.25}$$

The second term in the expression above requires a mixed, second order correction, which we ignore after linearization. Thus, we find

$$\frac{\partial P_i}{\partial \mathcal{E}_j} = \frac{1}{(2\pi)^3}\mathrm{Im}\sum_{n=1}^{N_{\mathrm{el}}/2}\int_{\mathrm{BZ}} d\boldsymbol{k}\,\langle \psi_{n\boldsymbol{k}}^{\mathcal{E}_j}|\psi_{n\boldsymbol{k}}^{k_i}\rangle. \tag{3.26}$$

Therefore, to calculate the electronic susceptibility we need to calculate how the Kohn-Sham orbitals change with respect to their wavevector $\boldsymbol{k}$, and an electric field $\boldsymbol{\mathcal{E}}$. The derivatives with respect to wavevector can be found by finite differences [47] or the Sternheimer equation. The first order correction to the orbitals with respect to an electric field can be found with the Sternheimer equation. The final derivative of the electronic susceptibility with respect to atomic displacement can also be found using DFPT, although this is a third order correction to the energy. However, only the first order corrections to the Kohn-Sham orbitals are needed for third order energy corrections due to the $2n+1$ theorem [48]. The derivation of this theorem is outlined in Section 3.B. An alternative to DFPT is to use finite differences via

$$\frac{\partial \chi_{jk}}{\partial u_i} = \frac{1}{2\Delta}[\chi_{jk}(R_i+\Delta) - \chi_{jk}(R_i-\Delta)], \tag{3.27}$$

where Δ is a small atomic displacement.

3.3 Derivation of the Raman Spectra

In the previous section, we outlined how one can find the electronic contribution to the Raman spectra, which is how the electronic susceptibility changes with respect to an atomic displacement. In this section, we discuss the final missing piece that includes the interactions between phonons and photons. For this derivation, we follow Ref. [49]. We first consider the expression for the polarization

$$\boldsymbol{P} = \epsilon_0 \boldsymbol{\chi} \boldsymbol{\mathcal{E}} + \epsilon_0 \frac{\partial \boldsymbol{\chi}}{\partial \boldsymbol{u}} \boldsymbol{u} \cdot \boldsymbol{\mathcal{E}} + \dots, \tag{3.28}$$

where $\boldsymbol{\chi}$ has been written as an expansion with respect to atomic displacement to first order. The Hamiltonian of interest is

$$H = -\int_{\Omega} d\boldsymbol{r}\ \boldsymbol{P} \cdot \boldsymbol{\mathcal{E}}, \tag{3.29}$$

which leads to the interaction Hamiltonian

$$H_{\text{int}} = -\int_{\Omega} d\boldsymbol{r}\ \epsilon_0 \frac{\partial \boldsymbol{\chi}}{\partial \boldsymbol{u}} \boldsymbol{u} (\boldsymbol{\mathcal{E}} \cdot \boldsymbol{\mathcal{E}}). \tag{3.30}$$

In terms of quantum mechanical operators, the atomic displacement is written as

$$\boldsymbol{u}(\boldsymbol{r}) = \sum_{\boldsymbol{k}} \hat{\boldsymbol{\eta}}_{\boldsymbol{k}} \sqrt{\frac{\hbar}{2M\omega_k}} \left(c_{\boldsymbol{k}} e^{i\boldsymbol{k}\cdot\boldsymbol{r}} + c_{\boldsymbol{k}}^{\dagger} e^{-i\boldsymbol{k}\cdot\boldsymbol{r}} \right), \tag{3.31}$$

where $\hat{\boldsymbol{\eta}}_{\boldsymbol{k}}$ is the polarization direction of the atomic displacement, M is the mass associated with the phonon, $\omega_{\boldsymbol{k}}$ is the frequency associated with the wavevector $\boldsymbol{k}$, and $c_{\boldsymbol{k}}/c_{\boldsymbol{k}}^{\dagger}$ are annihliation/creation operators for phonons with a wavevector $\boldsymbol{k}$. Similarily, the electric field can be written in terms of quantum mechanical operators as

$$\boldsymbol{\mathcal{E}}(\boldsymbol{r}) = -i \sum_{\boldsymbol{p}} \hat{\boldsymbol{\eta}}_{\boldsymbol{p}} \sqrt{\frac{\hbar\omega_{\boldsymbol{p}}}{2\epsilon_0 \Omega}} \left(a_{\boldsymbol{p}} e^{i\boldsymbol{p}\cdot\boldsymbol{r}} - a_{\boldsymbol{p}}^{\dagger} e^{-i\boldsymbol{p}\cdot\boldsymbol{r}} \right), \tag{3.32}$$

where $\hat{\boldsymbol{\eta}}_{\boldsymbol{p}}$ is the polarization direction of the electric field, $\omega_{\boldsymbol{p}}$ is the frequency associated with the wavevector $\boldsymbol{p}$, and $a_{\boldsymbol{p}}/a_{\boldsymbol{p}}^{\dagger}$ are annihilation/creation operators for photons with a

wavevector $\boldsymbol{p}$. Inserting these two expressions into the interaction Hamiltonian we find

$$\begin{aligned}
H_{\text{int}} &= \int_{\Omega} d\boldsymbol{r}\ \epsilon_0 \frac{\partial \boldsymbol{\chi}}{\partial \boldsymbol{u}} \sum_{\boldsymbol{k}} \hat{\boldsymbol{\eta}}_{\boldsymbol{k}} \sqrt{\frac{\hbar}{2M\omega_k}} \left(c_{\boldsymbol{k}} e^{i\boldsymbol{k}\cdot\boldsymbol{r}} + c_{\boldsymbol{k}}^{\dagger} e^{-i\boldsymbol{k}\cdot\boldsymbol{r}} \right) \\
&\quad \times \sum_{\boldsymbol{p},\boldsymbol{p}'} \hat{\boldsymbol{\eta}}_{\boldsymbol{p}} \cdot \hat{\boldsymbol{\eta}}_{\boldsymbol{p}'} \frac{\hbar\sqrt{\omega_{\boldsymbol{p}}\omega_{\boldsymbol{p}'}}}{2\omega_0\Omega} \left(a_{\boldsymbol{p}} e^{i\boldsymbol{p}\cdot\boldsymbol{r}} - a_{\boldsymbol{p}}^{\dagger} e^{-i\boldsymbol{p}\cdot\boldsymbol{r}} \right) \left(a_{\boldsymbol{p}'} e^{i\boldsymbol{p}'\cdot\boldsymbol{r}} - a_{\boldsymbol{p}'}^{\dagger} e^{-i\boldsymbol{p}'\cdot\boldsymbol{r}} \right) \\
&= \int_{\Omega} d\boldsymbol{r}\ \epsilon_0 \frac{\partial \boldsymbol{\chi}}{\partial \boldsymbol{u}} \sum_{\boldsymbol{k},\boldsymbol{p},\boldsymbol{p}'} \hat{\boldsymbol{\eta}}_{\boldsymbol{k}} \left(\hat{\boldsymbol{\eta}}_{\boldsymbol{p}} \cdot \hat{\boldsymbol{\eta}}_{\boldsymbol{p}'} \right) \sqrt{\frac{\hbar}{2M\omega_k}} \frac{\hbar\sqrt{\omega_{\boldsymbol{p}}\omega_{\boldsymbol{p}'}}}{2\omega_0\Omega} \\
&\quad \times \Big(c_{\boldsymbol{k}} a_{\boldsymbol{p}} a_{\boldsymbol{p}'} e^{i(\boldsymbol{k}+\boldsymbol{p}+\boldsymbol{p}')\cdot\boldsymbol{r}} - c_{\boldsymbol{k}} a_{\boldsymbol{p}} a_{\boldsymbol{p}'}^{\dagger} e^{i(\boldsymbol{k}+\boldsymbol{p}-\boldsymbol{p}')\cdot\boldsymbol{r}} - c_{\boldsymbol{k}} a_{\boldsymbol{p}}^{\dagger} a_{\boldsymbol{p}'} e^{i(\boldsymbol{k}-\boldsymbol{p}+\boldsymbol{p}')\cdot\boldsymbol{r}} \\
&\quad + c_{\boldsymbol{k}} a_{\boldsymbol{p}}^{\dagger} a_{\boldsymbol{p}'}^{\dagger} e^{i(\boldsymbol{k}-\boldsymbol{p}-\boldsymbol{p}')\cdot\boldsymbol{r}} + c_{\boldsymbol{k}}^{\dagger} a_{\boldsymbol{p}} a_{\boldsymbol{p}'} e^{i(-\boldsymbol{k}+\boldsymbol{p}+\boldsymbol{p}')\cdot\boldsymbol{r}} - c_{\boldsymbol{k}}^{\dagger} a_{\boldsymbol{p}} a_{\boldsymbol{p}'}^{\dagger} e^{i(-\boldsymbol{k}+\boldsymbol{p}-\boldsymbol{p}')\cdot\boldsymbol{r}} \\
&\quad - c_{\boldsymbol{k}}^{\dagger} a_{\boldsymbol{p}}^{\dagger} a_{\boldsymbol{p}'} e^{i(-\boldsymbol{k}-\boldsymbol{p}+\boldsymbol{p}')\cdot\boldsymbol{r}} + c_{\boldsymbol{k}}^{\dagger} a_{\boldsymbol{p}}^{\dagger} a_{\boldsymbol{p}'}^{\dagger} e^{i(-\boldsymbol{k}-\boldsymbol{p}-\boldsymbol{p}')\cdot\boldsymbol{r}} \Big) .
\end{aligned} \tag{3.33}$$

The integral over the volume can be broken into a summation over the unit cells, where each term in the summation has an integration over the unit cell. In addition, we consider the case where the wavelength of light is much larger than the size of the unit cell. Therefore,

the $\boldsymbol{r}$ dependence is replaced with a lattice displacement $\boldsymbol{R}$ and we are left with

$$\begin{aligned}
H_{\text{int}} &= \frac{1}{N}\sum_{\boldsymbol{R}}\epsilon_0\frac{\partial\boldsymbol{\chi}}{\partial\boldsymbol{u}}\sum_{\boldsymbol{k},\boldsymbol{p},\boldsymbol{p}'}\hat{\boldsymbol{\eta}}_{\boldsymbol{k}}\left(\hat{\boldsymbol{\eta}}_{\boldsymbol{p}}\cdot\hat{\boldsymbol{\eta}}_{\boldsymbol{p}'}\right)\sqrt{\frac{\hbar}{2M\omega_k}}\frac{\hbar\sqrt{\omega_{\boldsymbol{p}}\omega_{\boldsymbol{p}'}}}{2\omega_0\Omega} \\
&\quad\times\Big(c_{\boldsymbol{k}}a_{\boldsymbol{p}}a_{\boldsymbol{p}'}e^{i(\boldsymbol{k}+\boldsymbol{p}+\boldsymbol{p}')\cdot\boldsymbol{R}} - c_{\boldsymbol{k}}a_{\boldsymbol{p}}a^\dagger_{\boldsymbol{p}'}e^{i(\boldsymbol{k}+\boldsymbol{p}-\boldsymbol{p}')\cdot\boldsymbol{R}} - c_{\boldsymbol{k}}a^\dagger_{\boldsymbol{p}}a_{\boldsymbol{p}'}e^{i(\boldsymbol{k}-\boldsymbol{p}+\boldsymbol{p}')\cdot\boldsymbol{R}} \\
&\quad+c_{\boldsymbol{k}}a^\dagger_{\boldsymbol{p}}a^\dagger_{\boldsymbol{p}'}e^{i(\boldsymbol{k}-\boldsymbol{p}-\boldsymbol{p}')\cdot\boldsymbol{R}} + c^\dagger_{\boldsymbol{k}}a_{\boldsymbol{p}}a_{\boldsymbol{p}'}\delta_{(-\boldsymbol{k}+\boldsymbol{p}+\boldsymbol{p}')\cdot\boldsymbol{R}} - c^\dagger_{\boldsymbol{k}}a_{\boldsymbol{p}}a^\dagger_{\boldsymbol{p}'}e^{i(-\boldsymbol{k}+\boldsymbol{p}-\boldsymbol{p}')\cdot\boldsymbol{R}} \\
&\quad-c^\dagger_{\boldsymbol{k}}a^\dagger_{\boldsymbol{p}}a_{\boldsymbol{p}'}e^{i(-\boldsymbol{k}-\boldsymbol{p}+\boldsymbol{p}')\cdot\boldsymbol{R}} + c^\dagger_{\boldsymbol{k}}a^\dagger_{\boldsymbol{p}}a^\dagger_{\boldsymbol{p}'}e^{i(-\boldsymbol{k}-\boldsymbol{p}-\boldsymbol{p}')\cdot\boldsymbol{R}}\Big) \\
&= \epsilon_0\frac{\partial\boldsymbol{\chi}}{\partial\boldsymbol{u}}\sum_{\boldsymbol{k},\boldsymbol{p},\boldsymbol{p}'}\hat{\boldsymbol{\eta}}_{\boldsymbol{k}}\left(\hat{\boldsymbol{\eta}}_{\boldsymbol{p}}\cdot\hat{\boldsymbol{\eta}}_{\boldsymbol{p}'}\right)\sqrt{\frac{\hbar}{2M\omega_k}}\frac{\hbar\sqrt{\omega_{\boldsymbol{p}}\omega_{\boldsymbol{p}'}}}{2\omega_0\Omega} \\
&\quad\times\Big(c_{\boldsymbol{k}}a_{\boldsymbol{p}}a_{\boldsymbol{p}'}\delta_{\boldsymbol{p},-\boldsymbol{k}-\boldsymbol{p}'} - c_{\boldsymbol{k}}a_{\boldsymbol{p}}a^\dagger_{\boldsymbol{p}'}\delta_{\boldsymbol{p},-\boldsymbol{k}+\boldsymbol{p}'} - c_{\boldsymbol{k}}a^\dagger_{\boldsymbol{p}}a_{\boldsymbol{p}'}\delta_{\boldsymbol{p},\boldsymbol{k}+\boldsymbol{p}'} \\
&\quad+c_{\boldsymbol{k}}a^\dagger_{\boldsymbol{p}}a^\dagger_{\boldsymbol{p}'}\delta_{\boldsymbol{p},\boldsymbol{k}-\boldsymbol{p}'} + c^\dagger_{\boldsymbol{k}}a_{\boldsymbol{p}}a_{\boldsymbol{p}'}\delta_{\boldsymbol{p},\boldsymbol{k}-\boldsymbol{p}'} - c^\dagger_{\boldsymbol{k}}a_{\boldsymbol{p}}a^\dagger_{\boldsymbol{p}'}\delta_{\boldsymbol{p},\boldsymbol{k}+\boldsymbol{p}'} \\
&\quad-c^\dagger_{\boldsymbol{k}}a^\dagger_{\boldsymbol{p}}a_{\boldsymbol{p}'}\delta_{\boldsymbol{p},-\boldsymbol{k}+\boldsymbol{p}'} + c^\dagger_{\boldsymbol{k}}a^\dagger_{\boldsymbol{p}}a^\dagger_{\boldsymbol{p}'}\delta_{\boldsymbol{p},-\boldsymbol{k}-\boldsymbol{p}'}\Big) \\
&= \epsilon_0\frac{\partial\boldsymbol{\chi}}{\partial\boldsymbol{u}}\sum_{\boldsymbol{k},\boldsymbol{p}'}\hat{\boldsymbol{\eta}}_{\boldsymbol{k}}\left(\hat{\boldsymbol{\eta}}_{\boldsymbol{p}'}\cdot\hat{\boldsymbol{\eta}}_{\boldsymbol{p}'}\right)\sqrt{\frac{\hbar}{2M\omega_k}}\frac{\hbar\omega_{\boldsymbol{p}'}}{2\omega_0\Omega} \\
&\quad\times\Big(c_{\boldsymbol{k}}a_{-\boldsymbol{k}-\boldsymbol{p}'}a_{\boldsymbol{p}'} - c_{\boldsymbol{k}}a_{-\boldsymbol{k}+\boldsymbol{p}'}a^\dagger_{\boldsymbol{p}'} - c_{\boldsymbol{k}}a^\dagger_{\boldsymbol{k}+\boldsymbol{p}'}a_{\boldsymbol{p}'} \\
&\quad+c_{\boldsymbol{k}}a^\dagger_{\boldsymbol{k}-\boldsymbol{p}'}a^\dagger_{\boldsymbol{p}'} + c^\dagger_{\boldsymbol{k}}a_{\boldsymbol{k}-\boldsymbol{p}'}a_{\boldsymbol{p}'} - c^\dagger_{\boldsymbol{k}}a_{\boldsymbol{k}+\boldsymbol{p}'}a^\dagger_{\boldsymbol{p}'} \\
&\quad-c^\dagger_{\boldsymbol{k}}a^\dagger_{\boldsymbol{k}-\boldsymbol{p}'}a_{\boldsymbol{p}'} + c^\dagger_{\boldsymbol{k}}a^\dagger_{-\boldsymbol{k}-\boldsymbol{p}'}a^\dagger_{\boldsymbol{p}'}\Big).
\end{aligned} \tag{3.34}$$

We now take this interaction Hamiltonian and use Fermi's golden rule

$$\frac{1}{\tau} = \frac{2\pi}{\hbar}\sum_f |\langle f|H_{\text{int}}|i\rangle|^2\delta(E_f - E_i) \tag{3.35}$$

where it can be shown that (ignoring polarization)

$$\frac{1}{\tau} = \frac{1}{2\pi}\left|\frac{\partial\boldsymbol{\chi}}{\partial\boldsymbol{u}}\right|^2\frac{\hbar}{2\rho\omega_{\boldsymbol{k}}}\frac{\omega^4_{\boldsymbol{q}'}}{(c/n)^3}(1+N_{\boldsymbol{k}}), \tag{3.36}$$

where ρ is the mass density of the phonons, c is the speed of light, and n is the index of refraction. This formula gives the scattering rate for a Stokes process. A similar derivation can be done for the Anti-Stokes process. In addition to Stokes and Anti-Stokes processes, there may also be processes where phonons are not involved at all, which is called Rayleigh scattering. In this case the frequency of the outgoing light is the same as the incoming light.

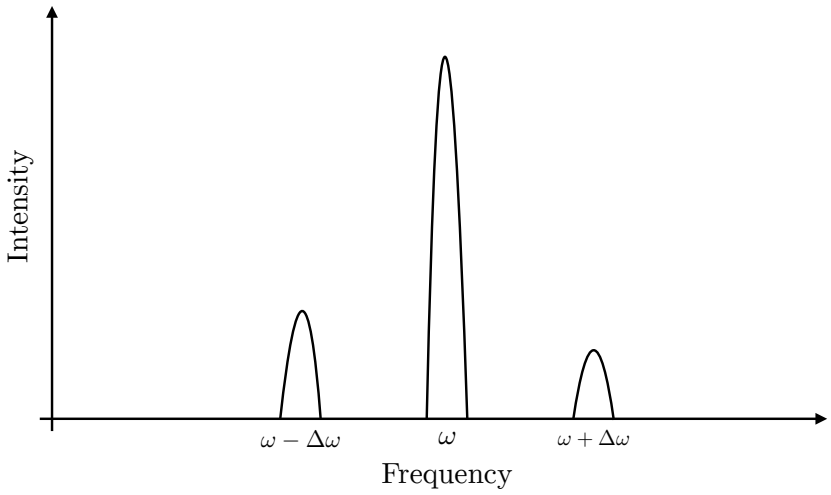


Figure 3.2: Illustration of a typical Raman spectra. The largest intensity is at ω, which corresponds to Rayleigh scattering. The second largest intensity comes from Stokes scattering, and the smallest intensity from Anti-Stokes scattering.

All three processes are shown in Figure 3.2. When taking into account the susceptibility tensor for a crystal in powder form, one must average over the possible directions [50, 51]. This yields the expression given in Chapter 10.

3.A Multi-parameter Perturbative Expansion

Here, we consider how to deal with a two parameter expansion in perturbation theory. The general expansion for the energy with respect to a parameter λ_1 and a parameter λ_2 is

$$\begin{aligned} E &= E^{(0)} \\ &+\lambda_1 \frac{\partial E}{\partial \lambda_1} + \lambda_2 \frac{\partial E}{\partial \lambda_2} \\ &+\frac{\lambda_1 \lambda_2}{2}\left(\frac{\partial^2 E}{\partial \lambda_1 \partial \lambda_2} + \frac{\partial^2 E}{\partial \lambda_2 \partial \lambda_1}\right) + \frac{\lambda_1^2}{2}\frac{\partial^2 E}{\partial \lambda_1^2} + \frac{\lambda_2^2}{2}\frac{\partial^2 E}{\partial \lambda_2^2} \\ &+\frac{\lambda_1^2 \lambda_2}{6}\frac{\partial^3 E}{\partial \lambda_1^2 \partial \lambda_2} + \frac{\lambda_1 \lambda_2^2}{6}\frac{\partial^3 E}{\partial \lambda_1 \partial \lambda_2^2} + \frac{\lambda_1^3}{6}\frac{\partial^3 E}{\partial \lambda_1^3} + \frac{\lambda_2^3}{6}\frac{\partial^3 E}{\partial \lambda_2^3} + \ldots \end{aligned} \tag{3.37}$$

The mixed derivatives are needed for the computation of the electronic susceptibility with respect to atomic displacement, for example. Instead of using finite differences, one can use

DFPT and the $2n+1$ theorem for the computation of mixed derivatives.

3.B The $2n+1$ Theorem

Introduced by Gonze *et al.* [48], the $2n+1$ theorem allows one to compute third order energy corrections with first order corrections to the Kohn-Sham orbitals. We will go through the proof to obtain the third order energy, but the process can be extended to any order. To begin, we first write the time-independent Schrödinger equation and normalization condition

$$\begin{aligned}(H^0-\epsilon^0)+\lambda(H^1-\epsilon^1)+\lambda^2(H^2-\epsilon^2)+\lambda^3(H^3-\epsilon^3)& \\ |\psi^0+\lambda\psi^1+\lambda^2\psi^2+\lambda^3\psi^3\rangle &= 0 \\ \langle\psi^0+\lambda\psi^1+\lambda^2\psi^2+\lambda^3\psi^3|\psi^0+\lambda\psi^1+\lambda^2\psi^2+\lambda^3\psi^3\rangle &= 1 \end{aligned} \tag{3.38}$$

We now sandwich the Hamiltonian expansion and consider the orders. At the zeroth order, we find

$$(H^0-\epsilon^0)|\psi^0\rangle = 0 \tag{3.39}$$

$$\langle\psi^0|\psi^0\rangle = 1 \tag{3.40}$$

which is just the usual time-independent Schrödinger equation and normalization condition. To first order we find

$$(H^1-\epsilon^1)|\psi^0\rangle+(H^0-\epsilon^0)|\psi^1\rangle = 0 \tag{3.41}$$

$$\langle\psi^1|\psi^0\rangle+\langle\psi^0|\psi^1\rangle = 0, \tag{3.42}$$

but since H^0 is assumed to be hermitian, after multiplying by $\langle\psi^0|$ we end up with

$$\langle\psi^0|(H^1-\epsilon^1)|\psi^0\rangle = 0 \tag{3.43}$$

which is the first order correction to the energy. It should be noted that because $\langle\psi^0|\psi^0\rangle = 1$, all other orders of the normalization must be 0. This is the usual convention. To second

order we have

$$(H^2 - \epsilon^2)|\psi^0\rangle + (H^0 - \epsilon^0)|\psi^2\rangle + (H^1 - \epsilon^1)|\psi^1\rangle = 0 \quad (3.44)$$

$$\langle\psi^0|\psi^2\rangle + \langle\psi^2|\psi^0\rangle + \langle\psi^1|\psi^1\rangle = 0 \quad (3.45)$$

which, after multiplying by $\langle\psi^0|$ can be simplified to

$$\langle\psi^0|(H^2 - \epsilon^2)|\psi^0\rangle + \langle\psi^0|(H^1 - \epsilon^1)|\psi^1\rangle = 0. \quad (3.46)$$

Normally, when one does perturbation theory to find a first order correction it is presented in a way where there is only a first order term in the perturbative Hamiltonian. To second order, the energy correction comes from the second term in the above equation. In general, if there is a second order term in the perturbative Hamiltonian then you also have the addition term $\langle\psi^0|H^2|\psi^0\rangle$ contributing to the second order energy. To third order we have

$$(H^0 - \epsilon^0)|\psi^3\rangle + (H^1 - \epsilon^1)|\psi^2\rangle + (H^2 - \epsilon^2)|\psi^1\rangle + (H^3 - \epsilon^3)|\psi^0\rangle = 0 \quad (3.47)$$

$$\langle\psi^0|\psi^3\rangle + \langle\psi^1|\psi^2\rangle + \langle\psi^2|\psi^1\rangle + \langle\psi^3|\psi^0\rangle = 0 \quad (3.48)$$

which can be simplified to

$$\begin{aligned} 0 &= \langle\psi^0|(H^1 - \epsilon^1)|\psi^2\rangle + \langle\psi^0|(H^2 - \epsilon^2)|\psi^1\rangle + \langle\psi^0|(H^3 - \epsilon^3)|\psi^0\rangle \\ \epsilon^3 &= \langle\psi^0|H^3|\psi^0\rangle - \langle\psi^1|(H^0 - \epsilon^0)|\psi^2\rangle + \langle\psi^0|(H^2 - \epsilon^2)|\psi^1\rangle \\ &= \langle\psi^0|H^3|\psi^0\rangle + \langle\psi^1|(H^2 - \epsilon^2)|\psi^0\rangle + \langle\psi^1|(H^1 - \epsilon^1)|\psi^1\rangle + \langle\psi^0|(H^2 - \epsilon^2)|\psi^1\rangle \\ &= \langle\psi^0|H^3|\psi^0\rangle + \langle\psi^1|H^2|\psi^0\rangle + \langle\psi^1|(H^1 - \epsilon^1)|\psi^1\rangle + \langle\psi^0|H^2|\psi^1\rangle \end{aligned} \quad (3.49)$$

using various substitutions from expressions in previous orders. We can see here that the third order correction to the energy only depends on the first order energies and wavefunctions. This is precisely the $2n + 1$ theorem.

Now that we have seen the case for one perturbation, we must consider 6 perturbative terms per atom in our unit cell. 3 of the perturbative terms describe the atomic displacement

and the other 3 describe the electric field. The time-independent Schrödinger equation and normalization condition is

$$(H^0 - \epsilon^0) + \sum_{i=1}^{6} \lambda_i (H^{\lambda_i} - \epsilon^{\lambda_i}) + \sum_{i,j=1}^{6} \lambda_i \lambda_j (H^{\lambda_i \lambda_j} - \epsilon^{\lambda_i \lambda_j}) + \sum_{i,j,k=1}^{6} \lambda_i \lambda_j \lambda_k (H^{\lambda_i \lambda_j \lambda_k} - \epsilon^{\lambda_i \lambda_j \lambda_k})$$
$$\times \left| \psi^0 + \sum_{i=1}^{6} \lambda_i \psi^{\lambda_i} + \sum_{i,j=1}^{6} \lambda_i \lambda_j \psi^{\lambda_i \lambda_j} + \sum_{i,j,k=1}^{6} \lambda_i \lambda_j \lambda_k \psi^{\lambda_i \lambda_j \lambda_k} \right\rangle = 0 \tag{3.50}$$

$$\left\langle \psi^0 + \sum_{i=1}^{6} \lambda_i \psi^{\lambda_i} + \sum_{i,j=1}^{6} \lambda_i \lambda_j \psi^{\lambda_i \lambda_j} + \sum_{i,j,k=1}^{6} \lambda_i \lambda_j \lambda_k \psi^{\lambda_i \lambda_j \lambda_k} \right|$$
$$\times \left| \psi^0 + \sum_{i=1}^{6} \lambda_i \psi^{\lambda_i} + \sum_{i,j=1}^{6} \lambda_i \lambda_j \psi^{\lambda_i \lambda_j} + \sum_{i,j,k=1}^{6} \lambda_i \lambda_j \lambda_k \psi^{\lambda_i \lambda_j \lambda_k} \right\rangle = 1 \tag{3.51}$$

Once again, we find to zeroth order

$$(H^0 - \epsilon^0)|\psi^0\rangle = 0 \tag{3.52}$$
$$\langle \psi^0 | \psi^0 \rangle = 1. \tag{3.53}$$

To first order, we find

$$\sum_{i=1}^{6} \lambda_i (H^{\lambda_i} - \epsilon^{\lambda_i})|\psi^0\rangle + (H^0 - \epsilon^0) \sum_{i=1}^{6} \lambda_i |\psi^{\lambda_i}\rangle = 0 \tag{3.54}$$
$$\langle \psi^0 | \sum_{i=1}^{6} \lambda_i |\psi^{\lambda_i}\rangle + \sum_{i=1}^{6} \lambda_i \langle \psi^{\lambda_i} | \psi^0 \rangle = 0. \tag{3.55}$$

in which case we find that

$$\epsilon^{\lambda_i} = \langle \psi^0 | H^{\lambda^i} | \psi^0 \rangle. \tag{3.56}$$

To second order, we find

$$\sum_{i,j=1}^{6} \lambda_i \lambda_j (H^{\lambda_i \lambda_j} - \epsilon^{\lambda_i \lambda_j})|\psi^0\rangle$$
$$+ \sum_{i=1}^{6} \lambda_i (H^{\lambda_i} - \epsilon^{\lambda_i}) \sum_{j=1}^{6} \lambda_j |\psi^{\lambda_j}\rangle + (H^0 - \epsilon^0) \sum_{i,j=1}^{6} \lambda_i \lambda_j |\psi^{\lambda_i \lambda_j}\rangle = 0 \tag{3.57}$$
$$\langle \psi^0 | \sum_{i,j=1}^{6} \lambda_i \lambda_j |\psi^{\lambda_i \lambda_j}\rangle + \sum_{i,j=1}^{6} \lambda_i \lambda_j \langle \psi^{\lambda_i \lambda_j} | \psi^0 \rangle + \sum_{i=1}^{6} \lambda_i \langle \psi^{\lambda_i} | \sum_{j=1}^{6} \lambda_j |\psi^{\lambda_j}\rangle = 0 \tag{3.58}$$

which means that the second order energies are

$$\epsilon^{\lambda_i\lambda_j} = \langle\psi^0|H^{\lambda_i\lambda_j}|\psi^0\rangle + \langle\psi^0|(H^{\lambda_i} - \epsilon^{\lambda_i})|\psi^{\lambda_j}\rangle. \tag{3.59}$$

Lastly, to third order we have

$$\begin{aligned}
&\sum_{i,j,k=1}^{6} \lambda_i\lambda_j\lambda_k(H^{\lambda_i\lambda_j\lambda_k} - \epsilon^{\lambda_i\lambda_j\lambda_k})|\psi^0\rangle + (H^0 - \epsilon^0)\sum_{i,j,k=1}^{6} \lambda_i\lambda_j\lambda_k|\psi^{\lambda_i\lambda_j\lambda_k}\rangle \\
&+\sum_{i=1}^{6} \lambda_i(H^{\lambda_i} - \epsilon^{\lambda_i})\sum_{j,k=1}^{6} \lambda_j\lambda_k|\psi^{\lambda_j\lambda_k}\rangle + \sum_{i,j=1}^{6} \lambda_i\lambda_j(H^{\lambda_i\lambda_j} - \epsilon^{\lambda_i\lambda_j})\sum_{k=1}^{6} \lambda_k|\psi^{\lambda_k}\rangle = 0
\end{aligned} \tag{3.60}$$

$$\begin{aligned}
&\langle\psi^0\sum_{i,j,k=1}^{6} \lambda_i\lambda_j\lambda_k|\psi^{\lambda_i\lambda_j\lambda_k}\rangle + \sum_{i,j,k=1}^{6} \lambda_i\lambda_j\lambda_k\langle\psi^{\lambda_i\lambda_j\lambda_k}|\psi^0\rangle \\
&+\sum_{i=1}^{6} \lambda_i\langle\psi^{\lambda_i}|\sum_{j,k=1}^{6} \lambda_j\lambda_k|\psi^{\lambda_j\lambda_k}\rangle + \sum_{i,j=1}^{6} \lambda_i\lambda_j\langle\psi^{\lambda_i\lambda_j}|\sum_{k=1}^{6} \lambda_k|\psi^{\lambda_k}\rangle = 0
\end{aligned} \tag{3.61}$$

which means that the third energy is

$$\begin{aligned}
\sum_{i,j,k=1}^{6} \epsilon^{\lambda_i\lambda_j\lambda_k} &= \sum_{i,j,k=1}^{6} \langle\psi^0|H^{\lambda_i\lambda_j\lambda_k}|\psi^0\rangle + \langle\psi^0|(H^{\lambda_i} - \epsilon^{\lambda_i})|\psi^{\lambda_j\lambda_k}\rangle + \langle\psi^0|(H^{\lambda_i\lambda_j} - \epsilon^{\lambda_i\lambda_j})|\psi^{\lambda_k}\rangle \\
&= \sum_{i,j,k=1}^{6} \langle\psi^0|H^{\lambda_i\lambda_j\lambda_k}|\psi^0\rangle - \langle\psi^{\lambda_i}|(H^0 - \epsilon^0)|\psi^{\lambda_j\lambda_k}\rangle + \langle\psi^0|(H^{\lambda_i\lambda_j} - \epsilon^{\lambda_i\lambda_j})|\psi^{\lambda_k}\rangle \\
&= \sum_{i,j,k=1}^{6} \langle\psi^0|H^{\lambda_i\lambda_j\lambda_k}|\psi^0\rangle + \langle\psi^{\lambda_i}|(H^{\lambda_j\lambda_k} - \epsilon^{\lambda_j\lambda_k})|\psi^0\rangle \\
&\quad +\langle\psi^{\lambda_i}|(H^{\lambda_j} - \epsilon^{\lambda_j})|\psi^{\lambda_k}\rangle + \langle\psi^0|(H^{\lambda_i\lambda_j} - \epsilon^{\lambda_i\lambda_j})|\psi^{\lambda_k}\rangle \\
\epsilon^{\lambda_i\lambda_j\lambda_k} &= \langle\psi^0|H^{\lambda_i\lambda_j\lambda_k}|\psi^0\rangle + \langle\psi^{\lambda_i}|H^{\lambda_j\lambda_k}|\psi^0\rangle \\
&\quad +\langle\psi^{\lambda_i}|(H^{\lambda_j} - \epsilon^{\lambda_j})|\psi^{\lambda_k}\rangle + \langle\psi^0|H^{\lambda_i\lambda_j}|\psi^{\lambda_k}\rangle
\end{aligned} \tag{3.62}$$

To obtain this expression, we used the same manipulations as for the single parameter perturbation. One must be careful when considering the indices. A similar expression to the expression shown above can be found in [52], but there is an additional part which corresponds with the expansion of the Kohn-Sham Hamiltonian.

Chapter 4

Quantum Monte Carlo

In this chapter we discuss Monte Carlo integration and introduce the main concepts when applying it to atomistic systems. The methodology discussed here is used in Chapter 11. For Section 4.2 and Section 4.3 we follow Ref. [53] and refer the reader to this and references therein for an even more in depth look at these methodologies.

4.1 Monte Carlo Integration

Monte Carlo (MC) integration is a very simple, yet powerful integration technique. It's usefulness is much more clear when dealing with high dimensional functions. Consider the function $f(x) = \sin(x)$ where $x \in [0, \pi]$. The integral of this function can be done analytically

$$\begin{aligned} F &= \int_0^\pi dx\ \sin(x) \\ &= -\cos(\pi) + \cos(0) \\ &= 2. \end{aligned} \tag{4.1}$$

Using Monte Carlo integration, we approximate the integral by writing

$$F = \int_0^\pi dx\ \sin(x) \approx \frac{\pi}{N} \sum_{i=1}^{N} \sin(x_i) \tag{4.2}$$

where $x_i \in [0, \pi]$ is a randomly drawn sample, and N is the total number of samples. As we increase the number of samples, we should expect to converge the MC result. After $N = 10^6$ samples, we get a value of $F = 1.99965$. For this simple function, it is more efficient to use a

grid-based numerical integration scheme rather than using Monte Carlo integration. For high dimensional functions, like a many-body wavefunction $\Psi(\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_{N_e})$, grid-based integration methods become infeasible to use due to the unfavourable exponential scaling. It becomes impossible to store the function in memory (or on disk).

We can also improve the efficiency of the MC integration by doing importance sampling. This avoids sampling parts of a function that contribute very little to the integral. We introduce a probability density function, $p(x)$, such that

$$1 = \int_0^\pi dx\, p(x). \tag{4.3}$$

where $p(x)$ is a good representation of the function we are trying to integrate, $f(x)$. We now choose N samples $x_1, x_2, \ldots x_N$ based on the probability distribution $p(x)$ using the Metropolis algorithm [54]. In the Metropolis algorithm, we can calculate a random walk with the following steps:

1. Choose a starting position x_0.
2. Generate a random change Δx, which yields a trial move $x_1 = x_0 + \Delta x$.
3. Accept the trial move with probability

$$P(x_0 \to x_1) = \min\left(1, \frac{p(x_1)}{p(x_0)}\right). \tag{4.4}$$

4. If accepted, set x_1 to x_0. Otherwise, reject the trial move. Go to step 2.

To calculate our integral, we weight the function by our probability function and evaluate both f and p from the positions generated from the random walk via

$$F \approx \frac{1}{N}\sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}. \tag{4.5}$$

4.2 Variational Monte Carlo

In variational Monte Carlo (VMC), one constructs an ansatz for the many-body wavefunction with parameters that can be altered to achieve a minimum energy. In our case, we construct the a trial, many-body wavefunction from Kohn-Sham orbitals along with a Jastrow factor

$$\begin{aligned}\Psi_{\mathrm{T}}(\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_{N_\mathrm{e}}) &= \frac{1}{\sqrt{N_\mathrm{e}}} \begin{vmatrix} \psi_1(\boldsymbol{r}_1) & \psi_1(\boldsymbol{r}_2) & \ldots & \psi_1(\boldsymbol{r}_{N_\mathrm{e}}) \\ \psi_2(\boldsymbol{r}_1) & \psi_2(\boldsymbol{r}_2) & \ldots & \psi_2(\boldsymbol{r}_{N_\mathrm{e}}) \\ \vdots & \vdots & \ldots & \vdots \\ \psi_{N_\mathrm{e}/2}(\boldsymbol{r}_1) & \psi_{N_\mathrm{e}/2}(\boldsymbol{r}_2) & \ldots & \psi_{N_\mathrm{e}/2}(\boldsymbol{r}_{N_\mathrm{e}}) \end{vmatrix} \\ &\times \exp[J(\{\lambda\}, \boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_{N_\mathrm{e}})], \end{aligned} \tag{4.6}$$

where $\{\lambda\}$ are a set of parameters that can be modified to minimize the energy. The goal of the Jastrow factor is to add in dynamic electron correlation through two-body terms

$$J_2(\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_{N_\mathrm{e}}) = \sum_{i=1}^{N_\mathrm{e}} \sum_{j \neq i}^{N_\mathrm{e}} u(\{\lambda\}, |\boldsymbol{r}_i - \boldsymbol{r}_j|). \tag{4.7}$$

There are also one-body terms that can (and should) be included to improve the electron density while introducing dynamic correlation [53]. They take the form

$$J_1(\boldsymbol{r}) = \sum_{i=1}^{N_\mathrm{ion}} u(\{\lambda\}, \boldsymbol{r} - \boldsymbol{R}_i). \tag{4.8}$$

With this ansatz the goal is to compute

$$E_\mathrm{T} = \frac{\int_\Omega d\boldsymbol{r}\ \Psi_\mathrm{T}^*(\boldsymbol{r}) H_\mathrm{e} \Psi_\mathrm{T}(\boldsymbol{r})}{\int_\Omega d\boldsymbol{r}\ |\Psi_\mathrm{T}(\boldsymbol{r})|^2} \geq E_0, \tag{4.9}$$

where $\boldsymbol{r} = (\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_{N_\mathrm{e}})$ is the $3N_\mathrm{e}$ dimensional space, H_e is the electronic Hamiltonian from 2.2, and E_0 is the true ground state energy. This expression can be transformed into a

more advantageous form by multiplying by $\Psi_{\mathrm{T}}\Psi_{\mathrm{T}}^{-1}$ yielding

$$E_{\mathrm{T}} = \frac{\int_{\Omega} d\boldsymbol{r}\ |\Psi_{\mathrm{T}}(\boldsymbol{r})|^2 \Psi_{\mathrm{T}}(\boldsymbol{r})^{-1} H_{\mathrm{e}} \Psi_{\mathrm{T}}(\boldsymbol{r})}{\int_{\Omega} d\boldsymbol{r}\ |\Psi_{\mathrm{T}}(\boldsymbol{r})|^2} \tag{4.10}$$

$$= \int_{\Omega} d\boldsymbol{r}\ P(\boldsymbol{r}) E_{\mathrm{L}}(\boldsymbol{r}), \tag{4.11}$$

where

$$P(\boldsymbol{r}) = \frac{|\Psi_{\mathrm{T}}(\boldsymbol{r})|^2}{\int_{\Omega} d\boldsymbol{r}\ |\Psi_{\mathrm{T}}(\boldsymbol{r})|^2} \tag{4.12}$$

is the probability distribution that we use to sample electron configurations, and

$$E_{\mathrm{L}}(\boldsymbol{r}) = \Psi_{\mathrm{T}}(\boldsymbol{r})^{-1} H_{\mathrm{e}} \Psi_{\mathrm{T}}(\boldsymbol{r}) \tag{4.13}$$

is defined to be the local energy. As done in Section 4.1, we sample the configuration space using the Metropolis algorithm and evaluate the integrand at the sampled points to obtain our value of energy via

$$E_{\mathrm{T}} \approx \frac{1}{N} \sum_{i=1}^{N} E_{\mathrm{L}}(\boldsymbol{r}_i). \tag{4.14}$$

Once the energy has been evaluated, the parameters $\{\lambda\}$ can be updated (via gradient descent, for example) to minimize the energy.

4.3 Diffusion Monte Carlo

4.3.1 Introduction and Fundamentals

Diffusion Monte Carlo (DMC) is an alternative, more accurate QMC method that projects out the ground state wavefunction using a time evolution operator. First, consider the time-dependent many-body Schrödinger equation (in atomic units)

$$i \frac{\partial \Psi(\boldsymbol{r}, t)}{\partial t} = H_{\mathrm{e}} \Psi(\boldsymbol{r}, t) \tag{4.15}$$

where H_{e} is the electronic Hamiltonian defined in Section 2.1, and $\boldsymbol{r}$ represents the $3N_{\mathrm{e}}$ coordinate space. The electronic Hamiltonian is not time-dependent and therefore the

solution can be written as

$$\Psi(\boldsymbol{r},t)=\sum_{n=1}^{\infty}c_n\psi_n(\boldsymbol{r})e^{-iE_nt}, \tag{4.16}$$

where ψ_n and E_n are found from the time-independent Schrödinger equation

$$H_{\mathrm{e}}\psi_n(\boldsymbol{r})=E_n\psi_n(\boldsymbol{r}). \tag{4.17}$$

Now consider the transformation from real time to imaginary time ($\tau = it$). The time-dependent Schrödinger equation becomes a diffusion equation and the solution changes to

$$\Psi(\boldsymbol{r},\tau)=\sum_{n=1}^{\infty}c_n\psi_n(\boldsymbol{r})e^{-E_n\tau}. \tag{4.18}$$

Consider now the subtraction of a constant energy to the electronic Hamiltonian E_T. This yields the solution

$$\Psi(\boldsymbol{r},\tau)=\sum_{n=1}^{\infty}c_n\psi_n(\boldsymbol{r})e^{-(E_n-E_T)\tau}. \tag{4.19}$$

In the asymptotic limit of $\tau\to\infty$ the solution is

$$\Psi(\boldsymbol{r},\tau)=\begin{cases}\infty & \text{if } E_T>E_0\\ 0 & \text{if } E_T<E_0\\ c_0\psi_0 & \text{if } E_T=E_0.\end{cases} \tag{4.20}$$

Therefore, if we carefully adjust E_T, we can obtain the ground state energy and wavefunction. In practice, this is done by time evolving an initial state

$$|\Psi(\tau)\rangle=U(\tau)|\Psi(0)\rangle, \tag{4.21}$$

where the time evolution operator is defined as

$$U(\tau)=e^{-\tau(H_{\mathrm{e}}-E_T)}. \tag{4.22}$$

If we now perform projections onto position space $\boldsymbol{r}$ and insert a complete set for $\boldsymbol{r}'$ we find

$$\begin{aligned} \langle \boldsymbol{r}|\Psi(\tau)\rangle &= \langle \boldsymbol{r}|U(\tau)\int d\boldsymbol{r}'\ |\boldsymbol{r}'\rangle\langle \boldsymbol{r}'|\Psi(0)\rangle \\ \Psi(\boldsymbol{r},\tau) &= \int d\boldsymbol{r}'\ \langle \boldsymbol{r}|U(\tau)|\boldsymbol{r}'\rangle\Psi(\boldsymbol{r}',0), \end{aligned} \tag{4.23}$$

where

$$G(\boldsymbol{r},\boldsymbol{r}',\tau) = \langle \boldsymbol{r}|U(\tau)|\boldsymbol{r}'\rangle. \tag{4.24}$$

$G(\boldsymbol{r},\boldsymbol{r}',\tau)$ is called a Green's function and it also obeys the imaginary time Schrödinger equation. It evolves, or propagates the wavefunction in both time and space. The Green's function is not analytically known, but is approximated in the following way. Using the Trotter-Suzuki formula for exponential sums of operators A and B

$$e^{-\tau(A+B)} = e^{-\tau B/2}e^{-\tau A}e^{-\tau B/2} + \mathcal{O}(\tau^3), \tag{4.25}$$

we can solve for the Green's function of each piece of our Hamiltonian separately. One is now limited in how far we can propagate in time due to the error proportional to $\mathcal{O}(\tau^3)$, and therefore must take many short steps for the limit $\tau \to \infty$. For the operator A, we assign it to the sum of the kinetic energy operators for the electrons

$$\partial_\tau G_T(\boldsymbol{r},\boldsymbol{r}',\tau) = T(\boldsymbol{r})G_T(\boldsymbol{r},\boldsymbol{r}',\tau) = \frac{1}{2}\sum_{i=1}^{N_\mathrm{e}} \nabla^2_{\boldsymbol{r}_i} G_T(\boldsymbol{r},\boldsymbol{r}',\tau) \tag{4.26}$$

and we see we are left with a diffusion equation in $3N_\mathrm{e}$ dimensional space, where the solution is a $3N_\mathrm{e}$ dimensional Gaussian

$$G_T(\boldsymbol{r},\boldsymbol{r}',\tau) = \prod_{i=1}^{N_\mathrm{e}} \frac{1}{(2\pi\tau)^{3/2}} e^{-(\boldsymbol{r}_i-\boldsymbol{r}_i')^2/2\tau}, \tag{4.27}$$

and where, in this case, $\boldsymbol{r}_i$ represents the coordinate space of electron i. To simplify the notation, we write

$$G_T(\boldsymbol{r},\boldsymbol{r}',\tau) = \langle \boldsymbol{r}|e^{-\tau T}|\boldsymbol{r}'\rangle = \frac{1}{(2\pi\tau)^{3N_\mathrm{e}/2}} e^{-(\boldsymbol{r}-\boldsymbol{r}')^2/2\tau}. \tag{4.28}$$

For the operator B, we assign it to the remaining part of the Hamiltonian that includes the interactions between particles $V(\boldsymbol{r})$ and the constant energy shift E_T. Putting everything together we find,

$$\begin{aligned} G(\boldsymbol{r},\boldsymbol{r}',\tau) &= \langle \boldsymbol{r}|e^{-\tau(T+V-E_T)}|\boldsymbol{r}'\rangle \\ &\approx e^{-\tau(V(\boldsymbol{r})-E_T)/2}\langle \boldsymbol{r}|e^{-\tau T}|\boldsymbol{r}'\rangle e^{-\tau(V(\boldsymbol{r}')-E_T)/2} \\ &\approx \frac{1}{(2\pi\tau)^{3N_e/2}} e^{-(\boldsymbol{r}-\boldsymbol{r}')^2/2\tau} e^{-\tau(V(\boldsymbol{r})+V(\boldsymbol{r}')-2E_T)/2}. \end{aligned} \tag{4.29}$$

This is the approximate Green's function for small τ. In practice, one defines *walkers* such that the state at $\tau = 0$ is

$$\Psi(\boldsymbol{R},0) = \sum_{j}^{N_{\text{walkers}}} \delta(\boldsymbol{r}-\boldsymbol{r}_j). \tag{4.30}$$

A birth/death process for each walker is defined based on the additional factor to the Green's function

$$e^{-\tau(V(\boldsymbol{r})+V(\boldsymbol{r}')-2E_T)/2}. \tag{4.31}$$

where the goal is to populate regions that contribute large amounts of energy. Therefore, the number of walkers can change during the simulation. Due to this, integration can become very inefficient if the number of walkers varies too much from step to step. One must introduce importance sampling to improve the efficiency. In addition, one always converges to a node-less Bosonic state, which makes this procedure useless for Fermionic systems. To fix both of these problems, a trial function Ψ_T is introduced. This trial function is assumed to have the correct nodes of the Fermionic wavefunction, and acceptance rules are put in place such that walkers can not cross from positive regions to negative ones, or vice versa. This is formally known as the *fixed-node approximation.* Instead of solving for the wavefunction, we solve for a new, mixed probability density function

$$f(\boldsymbol{r},\tau) = \Psi(\boldsymbol{r},\tau)\Psi_T(\boldsymbol{r}). \tag{4.32}$$

Plugging this into the imaginary time Schrödinger equation, we find

$$\begin{aligned} -\frac{\partial}{\partial\tau}\frac{f(\boldsymbol{r},\tau)}{\Psi_T(\boldsymbol{r})} &= (H_{\mathrm{e}}-E_T)\frac{f(\boldsymbol{r},\tau)}{\Psi_T(\boldsymbol{r})} \\ -\frac{\partial f(\boldsymbol{r},\tau)}{\partial\tau} &= -\frac{1}{2}\sum_{i=1}^{N_{\mathrm{e}}}\nabla_{\boldsymbol{r}_i}^2 f(\boldsymbol{r},\tau)+\sum_{i=1}^{N_{\mathrm{e}}}\nabla_{\boldsymbol{r}_i}\cdot[f(\boldsymbol{r},\tau)\Psi_T^{-1}(\boldsymbol{r})\nabla_{\boldsymbol{r}_i}\Psi_T(\boldsymbol{r})]+(E_L(\boldsymbol{r})-E_T)f(\boldsymbol{r},\tau). \end{aligned} \tag{4.33}$$

To derive this expression, we used

$$\nabla_{\boldsymbol{r}_i}\Psi_T^{-1}(\boldsymbol{r}) = -\frac{\nabla_{\boldsymbol{r}_i}\Psi_T(\boldsymbol{r})}{\Psi_T^2(\boldsymbol{r})}. \tag{4.34}$$

The local energy $E_L(\boldsymbol{r})$ is the same expression used in Section 4.2 and the second term is associated with *drift* with velocity

$$\boldsymbol{v}_D = \sum_{i=1}^{N_{\mathrm{e}}}\Psi_T^{-1}(\boldsymbol{r})\nabla_{\boldsymbol{r}_i}\Psi_T(\boldsymbol{r}), \tag{4.35}$$

which modifies the diffusive part of the previous Green's function G_T to

$$G_d(\boldsymbol{r},\boldsymbol{r}',\tau) = \frac{1}{(2\pi\tau)^{3N/2}}e^{-(\boldsymbol{r}-\boldsymbol{r}'-\tau\boldsymbol{v}_D(\boldsymbol{r}'))^2/2\tau}. \tag{4.36}$$

This additional term causes the walkers to drift towards regions where Ψ_T is large, which in turn causes the density of walkers to increase. The second part of the approximated Green's function, referred to as the *branching* Green's function is

$$G_b(\boldsymbol{r},\boldsymbol{r}',\tau) = e^{-\tau(E_L(\boldsymbol{r})+E_L(\boldsymbol{r}')-2E_T)/2}. \tag{4.37}$$

Using the local energy $E_L(\boldsymbol{r})$ instead of the potential energy significantly reduces the fluctuations of walkers. This is because the local energy is a good approximation to the ground state energy with a good trial wavefunction, and it remains roughly constant over the course of a simulation. All together, we have the Green's function

$$\tilde{G}(\boldsymbol{r},\boldsymbol{r},\tau) \approx G_d(\boldsymbol{r},\boldsymbol{r}',\tau)G_b(\boldsymbol{r},\boldsymbol{r}',\tau), \tag{4.38}$$

which can be used to solve for the probability density f at a later time τ via

$$f(\boldsymbol{r}, \tau + \delta\tau) = \int d\boldsymbol{r}' \tilde{G}(\boldsymbol{r}, \boldsymbol{r}', \delta\tau) f(\boldsymbol{r}, \tau). \tag{4.39}$$

By introducing f, we have simply performed a similarity transformation on the original Green's function, where the transformation matrix consists of Ψ_T along the diagonal.

With everything defined, how does one obtain observables of the system? For the ground state energy one has a choice between using the offset E_T or by using an approximate mixed estimator. For the energy offset, one updates it based on the average values of $E_L(\boldsymbol{r})$ from the walkers

$$E_T = \frac{1}{2}(\langle E_L(\boldsymbol{r}) + E_L(\boldsymbol{r}')\rangle) \tag{4.40}$$

such that $G_b \approx 1$. However, to control the population of walkers, one adds an additional term $\alpha(1 - N_j/N)$, where α is a small parameter, N is the desired number of walkers, and N_j is the current number of walkers. The energy offset then becomes

$$E_T = \frac{1}{2}(\langle E_L(\boldsymbol{r}) + E_L(\boldsymbol{r}')\rangle) + \alpha(1 - N_j/N). \tag{4.41}$$

For an observable that commutes with the Hamiltonian, the mixed estimator is defined to be

$$\begin{aligned} \langle A \rangle &= \frac{\langle \Psi_0 | A | \Psi_T \rangle}{\langle \Psi_0 | \Psi_T \rangle} \\ &= \lim_{\tau \to \infty} \frac{\int d\boldsymbol{r}\ f(\boldsymbol{r}, \tau) \Psi_T^{-1}(\boldsymbol{r}) A \Psi_T(\boldsymbol{r})}{\int d\boldsymbol{r}\ f(\boldsymbol{r}, \tau)}, \end{aligned} \tag{4.42}$$

which means that the energy can be calculated as

$$\begin{aligned} \langle E \rangle &= \lim_{\tau \to \infty} \frac{\int d\boldsymbol{r}\ f(\boldsymbol{r}, \tau) E_L(\boldsymbol{r})}{\int d\boldsymbol{r}\ f(\boldsymbol{r}, \tau)} \\ &\approx \frac{1}{N} \sum_{n=1}^{N} E_L(\boldsymbol{r}_n) \end{aligned} \tag{4.43}$$

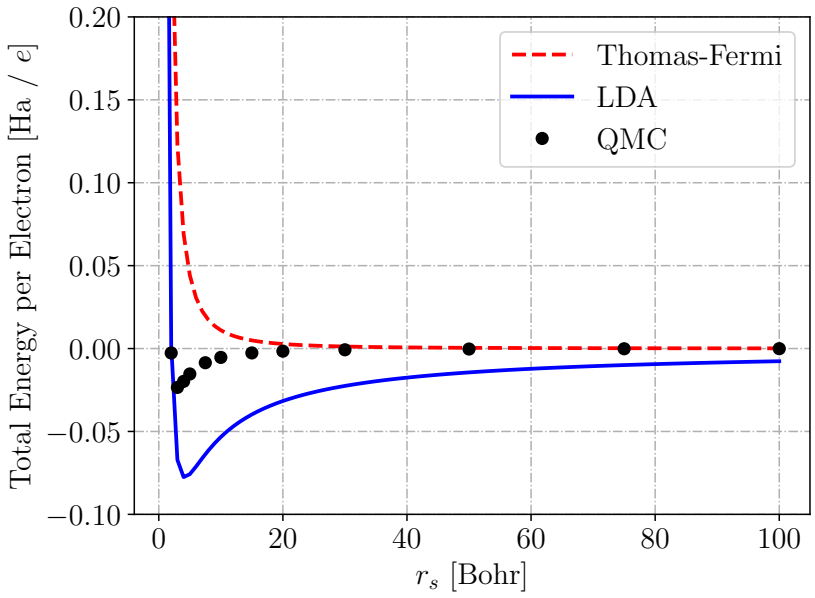


Figure 4.1: Total energy per electron of a free electron gas as a function of r_s. The Thomas-Fermi and LDA lines are taken from Figure 2.2. The QMC curve is taken from Ref. [55].

where $\boldsymbol{r}_n$ are sampled from the probability distribution $f(\boldsymbol{r}, \tau \to \infty)$. In practice, however, we can not sample directly from f, because detailed balance is not maintained due to the introduction of the drift velocity. The acceptance probability is modified as

$$P(\boldsymbol{r}, \boldsymbol{r}') = \min\left[1, \frac{f(\boldsymbol{r}', \tau)}{f(\boldsymbol{r}, \tau)}\right] \to \min\left[1, \frac{f(\boldsymbol{r}', \tau) G_d(\boldsymbol{r}', \boldsymbol{r}, \tau)}{f(\boldsymbol{r}, \tau) G_d(\boldsymbol{r}, \boldsymbol{r}', \tau)}\right]. \tag{4.44}$$

This methodology was used in Ref. [55] to calculate total internal energies of electron gasses. In Figure 4.1, we show these results along with results from the Thomas-Fermi model with and without LDA exchange-correlation energies. The QMC curve has a similar shape to the LDA curve for low values of r_s. However, the LDA curve is lower in energy.

4.3.2 The DMC Algorithm

The algorithm to perform DMC (Figure 4.2) is as follows:

1. Generate starting configurations for the walkers. The most optimal way to perform initialization is to sample the probability distribution $|\Psi_T|^2$. This is not mandatory, and

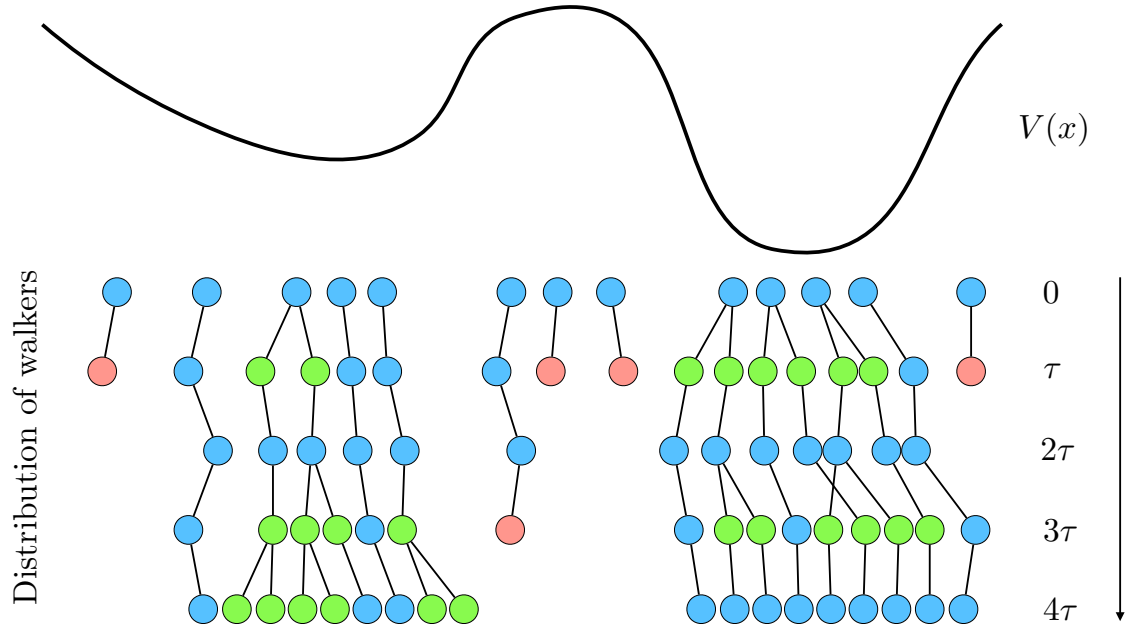


Figure 4.2: Visual progression of walkers during a diffusion Monte Carlo simulation, at time 0 the walkers are initialized according to some distribution. At each step in time walkers are deleted (red nodes), duplicated (green nodes), or left to continue on their random walk. As $\tau \to \infty$ the distribution of walkers converges to the ground state wavefunction.

one could start all of the walkers from the origin or randomly within the simulation cell.

2. For each electron coordinate within a walker, propose a move via

$$\boldsymbol{r} = \boldsymbol{r}' + \boldsymbol{\eta} + \tau \boldsymbol{v}_D(\boldsymbol{r}') \tag{4.45}$$

where $\boldsymbol{\eta}$ are (three) random numbers generated from a normal distribution with a mean of 0 and a standard deviation of $\sqrt{\tau}$.

3. Check if the walkers have changed sign. If they have, reject the move and continue.

4. Accept the new configuration according to Equation 4.44.

5. Based on the new configuration, remove, keep, or duplicate the walker according to $\text{int}(u + G_b(\boldsymbol{r}, \boldsymbol{r}, \tau))$.

6. Perform Steps 2-4 100-1000 times, which is called a block.

7. Update the offset energy E_T according to Equation 4.41.

8. Move forward in imaginary time and repeat from step 2 until convergence is reached.

CHAPTER 5

Machine Learning for Condensed Matter Physics

In condensed matter systems, we are always trying to calculate an observable of a system. This could be a number, vector, or tensor. For that reason, we focus on regression[1]. This chapter aims to introduce supervised machine learning to a physicist. We first discuss classic machine learning techniques and then move on to sophisticated, state-of-the-art deep learning techniques for regression problems.

5.1 Classic Machine Learning

5.1.1 Linear Regression

Let us start very simple with linear regression. Consider an ideal gas, and let's say we want to predict the pressure of the gas given a fixed volume and number of particles as a function of temperature $P(T)$. In our experimental data, we see a slight variation in the pressure measurements and therefore do not form a perfect line as we theoretically expect. In this case, our *feature vector* consists of one element, the temperature, and the output of our machine-learned function is the pressure. Our goal is to learn the mapping

$$\hat{P}_i(T_i) = aT_i + b, \tag{5.1}$$

[1]It should be noted that classification could be done if one has access to the full range of the predicted quantity. One can create a histogram of values and declare a value to be in one of the bins of the histogram. The only difference between classification and regression in machine learning is the loss function. The machinery of the algorithms remains the same for both tasks.

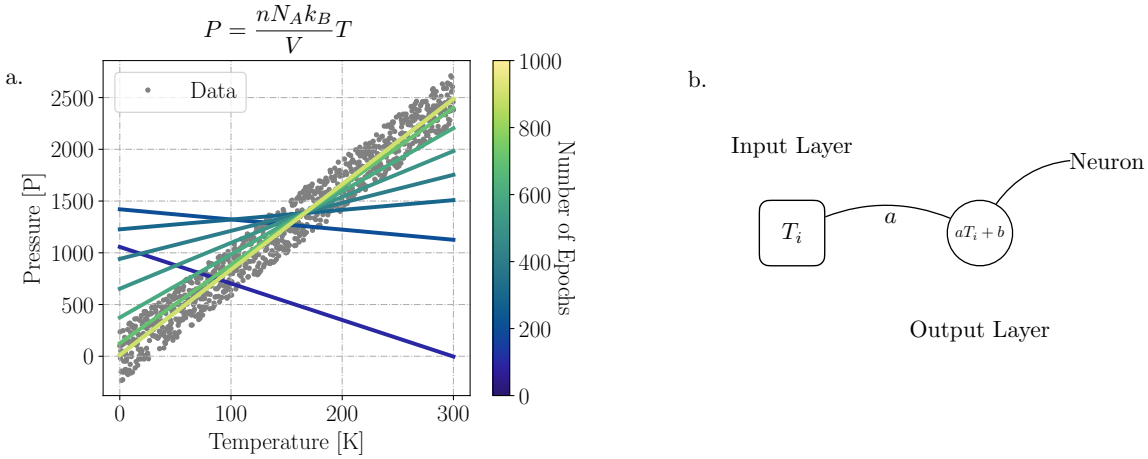


Figure 5.1: A toy example of linear regression using mini-batch gradient descent for an ideal gas. In (a) We can notice the improvement to the fit as gradient descent is being performed. In (b) we show the connection between linear regression and the formalism of machine learning.

which means we must solve for the coefficients a and b. Thus, at a constant gas density we are solving for the Boltzmann constant. T_i represents the true temperature and $\hat{P}_i$ is the predicted pressure given the temperature value of experiment i. To solve for a and b using gradient descent, we first initialize the variables a and b randomly, and pass all of the recorded temperatures into our function. We then consider the *loss* (or error) of our model, which can be defined by the mean squared error

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(P_i - \hat{P}_i)^2. \tag{5.2}$$

Here N is the total number of experiments, and P_i is the true pressure for experiment i. To find the values of a and b that minimize the value of this loss function, we update them by following the gradients

$$a = a - \alpha\frac{\partial L}{\partial a},\ b = b - \alpha\frac{\partial L}{\partial b}. \tag{5.3}$$

Here, α is a small, positive parameter which is called the *learning rate*. Evaluating the derivatives, we use chain rule

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial a} &= \frac{1}{N}\sum_{i=1}^{N}\frac{\partial \mathcal{L}}{\partial \hat{P}_i}\frac{\partial \hat{P}_i}{\partial a} \\ &= \frac{2}{N}\sum_{i=1}^{N}(P_i - \hat{P}_i)T_i \qquad (5.4)\\ \frac{\partial \mathcal{L}}{\partial b} &= \frac{1}{N}\sum_{i=1}^{N}\frac{\partial \mathcal{L}}{\partial \hat{P}_i}\frac{\partial \hat{P}_i}{\partial b} \\ &= \frac{2}{N}\sum_{i=1}^{N}(P_i - \hat{P}_i) \qquad (5.5)\end{aligned}$$

We then iterate, and continue to update our parameters until we reach some convergence criteria. In Figure 5.1, we show an example of linear regression. Here, we plot the fit as a function of number of *epochs* (number of gradient step updates in relation to dataset size) and we see that our fit becomes better. When we pass all of our data through the model while updating the parameters, that is defined to be 1 epoch. It is also worth noting that one can update the fitting parameters in batches rather than with all the examples at once. This is called mini-batch gradient descent. In the case of Figure 5.1, the total data set size was 1024, and the mini-batch size used was 16. Mini-batches are commonly used during training. There may be cases when an entire dataset cannot fit into memory[2]. Additionally, depending on the landscape in weight space, there is a particular batch size that is optimal for a given learning rate. However, in the case of large batch sizes, Ref. [56] has found that models typically converge to non-optimal local minima. On the contrary, using a batch size of 1 is called stochastic gradient descent. The stochastic nature comes in when selecting each training example.

Additionally, in Figure 5.1, we illustrate the connection between linear regression and neural network terminology. Our feature vector, or values of temperature, is known as the *input layer*. The output, or predicted pressure value, forms the *output layer*. The connection

[2]In the case where entire datasets are used to perform parameter updates, one refers to this as gradient descent rather than stochastic gradient descent or mini-batch gradient descent.

between the input layer and output layer is the *weight*, a, and the output element is called a *neuron*. The neuron receives input and returns an output that depends on the weighting of the particular connection.

5.2 Deep Learning

5.2.1 Artificial Neural Networks

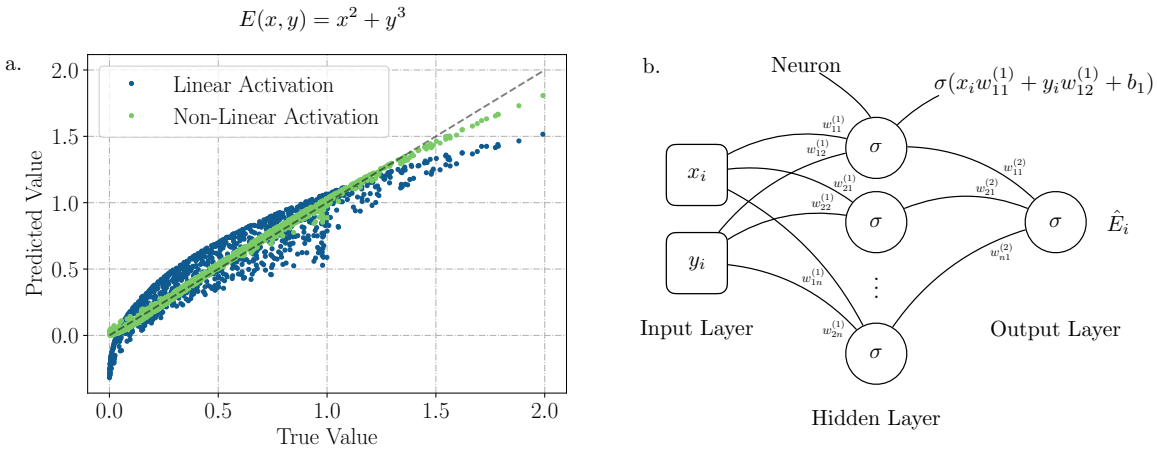


Figure 5.2: Predicted versus true (a) for a 2-dimensional non-linear function with and without non-linear activation functions. In (b) we show a diagram of a neural network with one hidden layer. See Section 5.2.1 for more explanation of the variables.

Linear regression is limited to problems where there there is a linear dependence between the input and output layers. Let's consider another toy problem of predicting a surface that depends on two spatial variables through the formula

$$E(x, y) = x^2 + y^3. \tag{5.6}$$

In this case, we know the surface is non-linear. There is an x-dependence that is quadratic and a y-dependence that is cubic. What can we add to our linear regression model to account for the non-linearity? Instead of going from our input layer directly to our output layer, let's add an intermediate layer, which is called a *hidden layer*. The input, hidden, and output layer form what is called an *artificial neural network* (ANN). Unfortunately, due to the mathematical structure of ANNs, we can only fit linear problems no matter how many

additional neurons we add. The key ingredient that we are missing is called an *activation function*. In the case of linear regression, the activation function was linear

$$\sigma(x) = x. \tag{5.7}$$

In machine learning, many activation functions exist. Two non-linear examples are the rectified linear unit (ReLU) [57]

$$\sigma(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{5.8}$$

and the exponential linear unit (ELU) [58]

$$\sigma(x) = \begin{cases} x & x > 0 \\ (e^{-x} - 1) & x \leq 0. \end{cases} \tag{5.9}$$

In Figure 5.2, we show predicted versus true results for ANNs with and without ELU activation functions. For a perfect model, all of the points lie along the diagonal line, shown in grey. With the inclusion of the non-linear activation functions, we can qualitatively see better performance. In this case, we used one layer consisting of 32 hidden neurons. The inputs $x, y \in [0, 1]$ were uniformly sampled. The networks were trained for 1000 epochs with a batch size of 16.

With the addition of the hidden layer comes the addition of new, trainable weights. To see how these weights are updated, let's consider a hidden layer with two neurons. The output vector of the 2 hidden neurons given an input vector $\boldsymbol{x}_i$ is written as

$$\boldsymbol{o}_i^{(1)} = \begin{pmatrix} \sigma(\boldsymbol{w}_1^{(1)} \boldsymbol{x}_i + b_1^{(1)}) \\ \sigma(\boldsymbol{w}_2^{(1)} \boldsymbol{x}_i + b_2^{(1)}) \end{pmatrix}, \tag{5.10}$$

and final output of the network is then calculated as

$$o_i^{(2)} \equiv \hat{E}_i = \boldsymbol{w}_1^{(2)} \boldsymbol{o}_i^{(1)} + b_1^{(2)}. \tag{5.11}$$

The input vector, represented by $\boldsymbol{x}_i = (x_i, y_i)$ is passed, or *forward propagated* through the ANN, and an output, $\hat{E}_i$ is computed. As discussed previously, a mini-batch of data with length N_{batch} is passed through the ANN, and a loss for that batch is computed via Equation 5.2. To update the weights, we again follow the gradients. For the weights and bias that connect the hidden layer to the output we find

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_{j1}^{(2)}} &= \frac{1}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} \frac{\partial \mathcal{L}}{\partial \hat{E}_i} \frac{\partial \hat{E}_i}{\partial w_{j1}^{(2)}} \\ &= \frac{2}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} (E_i - \hat{E}_i) o_{ij}^{(1)}, \end{aligned} \tag{5.12}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial b_{1}^{(2)}} &= \frac{1}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} \frac{\partial \mathcal{L}}{\partial \hat{E}_i} \frac{\partial \hat{E}_i}{\partial b_{1}^{(2)}} \\ &= \frac{2}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} (E_i - \hat{E}_i). \end{aligned} \tag{5.13}$$

For the weights and biases that connect the input layer to the hidden layer we consider the weights and biases associated with the input x_i

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_{j1}^{(1)}} &= \frac{1}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} \frac{\partial \mathcal{L}}{\partial \hat{E}_i} \frac{\partial \hat{E}_i}{\partial o_{ij}^{(1)}} \frac{\partial \sigma}{\partial w_{j1}^{(1)}} \\ &= \frac{2}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} (E_i - \hat{E}_i) w_{j1}^{(2)} \frac{\partial \sigma}{\partial w_{j1}^{(1)}}, \end{aligned} \tag{5.14}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial b_{1}^{(1)}} &= \frac{1}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} \frac{\partial \mathcal{L}}{\partial \hat{E}_i} \frac{\partial \hat{E}_i}{\partial o_{ij}^{(1)}} \frac{\partial \sigma}{\partial b_{1}^{(1)}} \\ &= \frac{2}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} (E_i - \hat{E}_i) w_{j1}^{(2)} \frac{\partial \sigma}{\partial b_{1}^{(1)}}. \end{aligned} \tag{5.15}$$

The derivatives of the activation function with respect to w or b are analytically known, and therefore can be implemented and evaluated rapidly to update all parameters. We can see that before evaluating changes of the variables that connect the input layer to the hidden layer, we need to first compute how the variables that connect the hidden layer to the output layer change. We move backwards through the network starting from the derivative of the loss function. This process is called *backpropagation*.

With the introduction of more neurons via hidden layers comes the question: How many neurons and layers does one need? The number of neurons, number of layers, choice of activation function, and learning rate in machine learning are called *hyperparameters*. One can either perform convergence tests as one of the hyperparameters changes or one can use Bayesian optimization techniques (i.e. Gaussian Process Regression) to find optimal parameters. In general, a hyperparameter search is computationally demanding since a model must be trained and evaluated for each hyperparameter choice.

There is also the question: How is deep learning different from machine learning? To answer this, one must first define what deep learning really means. According to Ref. [59], deep learning is simply using a network with many layers. The initial layers learn simple concepts, and the layers that follow use these simple concepts to make more complicated ones. Models with more layers require more data to find optimal parameters. With the influx of data worldwide, this gave rise to constructing large models via deep learning. The machinery may not change when considering machine learning and deep learning, but the number of trainable parameters do.

5.2.2 Additional Considerations for Training

Before moving on, we touch on other details that are needed during the training of the model. As discussed above, backpropagation is the key algorithm that allows us to update parameters within a network to acheive better predictions. In practice, however, one does not compute all of these derivatives manually. All of the derivatives are computed automatically after one forward pass of data. Machine learning packages keep track of the connections between all of the parameters, allowing one to efficiently compute all necessary derivatives. In addition, one does not just use gradient descent to update parameters, but more elaborate variants that include the parameter *momentum*. When using momentum, an additional term is added to Equation 5.3. The additional term comes from the previous update of a parameter multiplied by a value < 1. This is precisely what is implemented in the Adam optimizer [60]. If the gradient is in the same direction of the direction previously

travelled, we take larger update steps. Otherwise, the update steps are smaller, which signifies that we have found a local minima.

Another detail is weight initialization. One can not just simply start from all zeros due to the use of gradient descent methods. All derivatives (and therefore all subsequent updates) would be zero. The strategy taken is random initialization from a uniform distribution where the range of the parameters drawn depends on the number of input and output connections of the layer. This is referred to as Glorot initialization [61].

When training a model, there is a risk of *overfitting* the model parameters to the training data. With a dataset, a typical set up includes training, validation, and testing sets. Typically, the validation set is small compared to the training set. The test set is set aside until one has finished tuning their model. This is done so reported errors are an accurate estimate of the performance on unseen data. During training, errors are calculated on both the training and validation sets to monitor whether overfitting has occurred. If the error increases for the validation set but decreases for the training set, this is known as overfitting - the model is fitting to noise within the training set and no longer modelling a general trend. There are techniques, known as *regularization*, that can be used to avoid or decrease the amount of overfitting. Two similar techniques are called *L1 or L2 regularization*, where an additional constraint is placed on the model parameters such that the 1- or 2-norm of a parameter cannot become exceedingly large. Depending on the regularization choice we add the terms

$$\sum_{i}^{\text{all weights}} \beta|w_i|, \text{ or } \sum_{i}^{\text{all weights}} \gamma|w_i|^2 \tag{5.16}$$

to the loss function. Here, β and γ are small parameters. Another technique is called *dropout* [62], where neurons are randomly "turned off" during training. Both methods avoid overfitting by removing the possibility of having certain neurons dominating the output of a model.

Before performing training, there is also necessary pre-processing steps that must be done

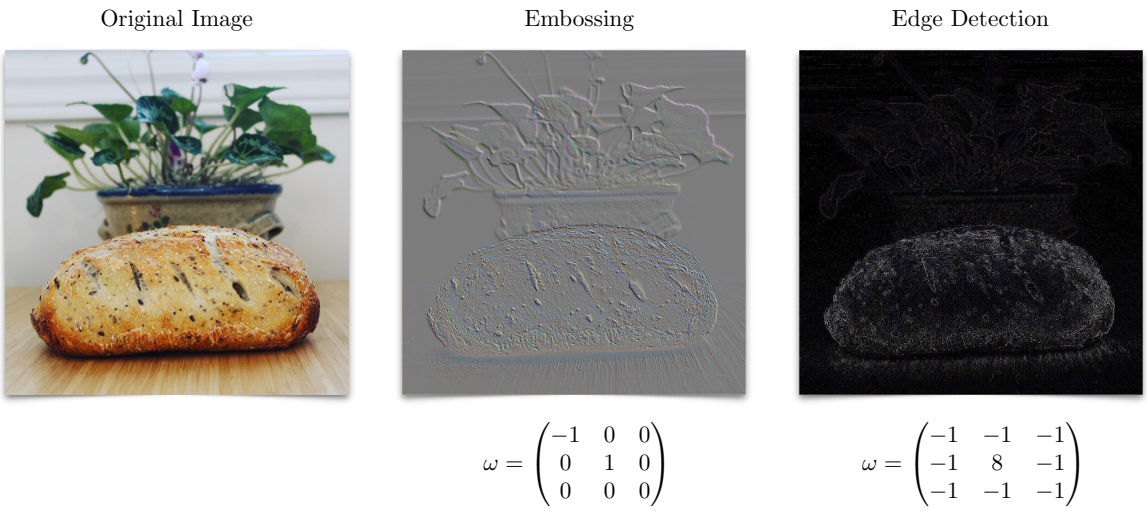


Figure 5.3: Examples of convolutional kernels being applied to an image. The image shown in the middle had an embossing kernel applied, which tends to extract colour differentials. The image on the right had an edge detection kernel applied, which extracts edges in the image.

on the dataset to ensure maximum performance. Two pre-processing techniques are data *standardization* or *normalization*. Some inputs may have vastly different ranges. Features with larger ranges will have much larger gradients, and training will become biased. To avoid this bias with standardization, one subtracts the mean and divides by the standard deviation for each feature in the dataset. This bias is less in the case of image-based data, but typically data standardization is still performed to improve training performance. Having a well-scaled dataset ensures that gradients are also well-scaled, allowing for maximum performance during the optimization. If the output of a network is a vector or tensor, where each element has different ranges, one can also perform normalization. Normalization is similar to standardization, but one subtracts the minimum value and divides by the range of values, such that all values are $\in [0, 1]$. Additionally, if using an activation function with a well defined range, one must transform their output data to fit this range.

5.2.3 Convolutional Neural Networks

ANNs work well when the input is 1-dimensional (a vector). What happens when the input is 2-dimensional, 3-dimensional, or higher? In image processing, features can be extracted

from images by performing a convolution with a specific *kernel*. How exactly are kernels applied? Let's consider a simple, 3×3, grey-scale image with values

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \tag{5.17}$$

and an embossing kernel

$$\omega = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{5.18}$$

as shown in Figure 5.3. The idea is to start at the top left of A, multiply our kernel by the respective elements, perform a summation, move to the next pixel, and repeat the operation. This process creates a new image. In order for this kernel to be applied, we must pad our image. A common form for *padding* in machine learning is by simply concatenating zeros around the image. This yields,

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 4 & 5 & 6 & 0 \\ 0 & 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{5.19}$$

Now let's apply the kernel. The first row of elements are

$$A_{11} * \omega = \begin{pmatrix} 0 \cdot -1 & 0 \cdot 0 & 0 \cdot 0 \\ 0 \cdot 0 & 1 \cdot 1 & 2 \cdot 0 \\ 0 \cdot 0 & 4 \cdot 0 & 5 \cdot 0 \end{pmatrix}, \tag{5.20}$$

$$A_{12} * \omega = \begin{pmatrix} 0 \cdot -1 & 0 \cdot 0 & 0 \cdot 0 \\ 1 \cdot 0 & 2 \cdot 1 & 3 \cdot 0 \\ 4 \cdot 0 & 5 \cdot 0 & 6 \cdot 0 \end{pmatrix}, \tag{5.21}$$

$$A_{13} * \omega = \begin{pmatrix} 0 \cdot -1 & 0 \cdot 0 & 0 \cdot 0 \\ 2 \cdot 0 & 3 \cdot 1 & 0 \cdot 0 \\ 5 \cdot 0 & 6 \cdot 0 & 0 \cdot 0 \end{pmatrix}. \tag{5.22}$$

Afterwards, we sum over the elements to yield the new pixels

$$B_{11} = \sum A_{11} * \omega = 1, \tag{5.23}$$

$$B_{12} = \sum A_{12} * \omega = 2, \tag{5.24}$$

$$B_{13} = \sum A_{13} * \omega = 3. \tag{5.25}$$

Performing this operation across all of the elements of A yields a new image B. It is important to note that kernel sizes can change, which means that the padding must also changed. Additionally, one can also apply a convolutional with a particular *stride*. If the stride is 1, then every pixel is visited during the convolution, and the resulting image is the same size. If the stride is 2, then we skip every other pixel, and size of the resultant image is halved in each dimension.

How are convolutions applied in machine learning? The weights within the kernels are also learnable parameters. Gradient descent can be applied to the convolutional kernels (and biases) in the exact same way as shown in Section 5.2.1. In addition, convolutions have built-in translational symmetry. One can shift an image in any direction, and a convolution of that image with a particular kernel will yield the same result. When one defines a con-

volutional layer in a neural network, one must choose the number of kernels, kernel size, padding, and stride. These choices are also hyperparameters that can be optimized. Neural networks that make use of convolutional filters are called convolutional neural networks (CNNs). It should be noted that convolutional neural networks can only learn non-local mappings when many layers are used. Typically, one uses many relatively small kernels (i.e. 5×5 in 2D) and therefore the resulting image from one convolutional layer is based on local information only (with a width of 2 pixels). However, when we apply another convolutional layer, information that was non-accessible when applying the first convolutional becomes available (total width now of 4 pixels). The more convolutional layers we have, the more non-local information becomes available. This is different from ANNs, where all of the information from the input is combined together to produce outputs. ANNs are inherently non-local.

A final note we make is about *channels*. Machine learning frameworks have been constructed to work on color images (which have 3 channels: red, blue, and green), since the original goals of CNNs were for detecting something in an image. In physics problems, where one is inputting a scalar field into a CNN, this is considered a grey-scale image. However, when one defines a convolutional layer with N kernels, each kernel produces a new image which collectively makes for an output with N channels. The introduction of channels allow one to upscale or downscale the amount of information that is being propagated through the network, and defines the number of learnable parameters within the convolutional layers.

5.2.4 Extensive Deep Neural Networks

Extensive deep neural networks (EDNNs) are an extension of DNNs that allows one to study systems with differing size. When using CNNs, one must define dimensions of inputs and outputs. These dimensions are locked when performing inference. As shown in Figure 5.4a, with EDNNs an input image is decomposed into sub-images, and the sub-images are fed into a machine learning model. The final value comes from the summation over all of the outputs from the sub-images. Each sub-image, therefore, gives a contribution to the

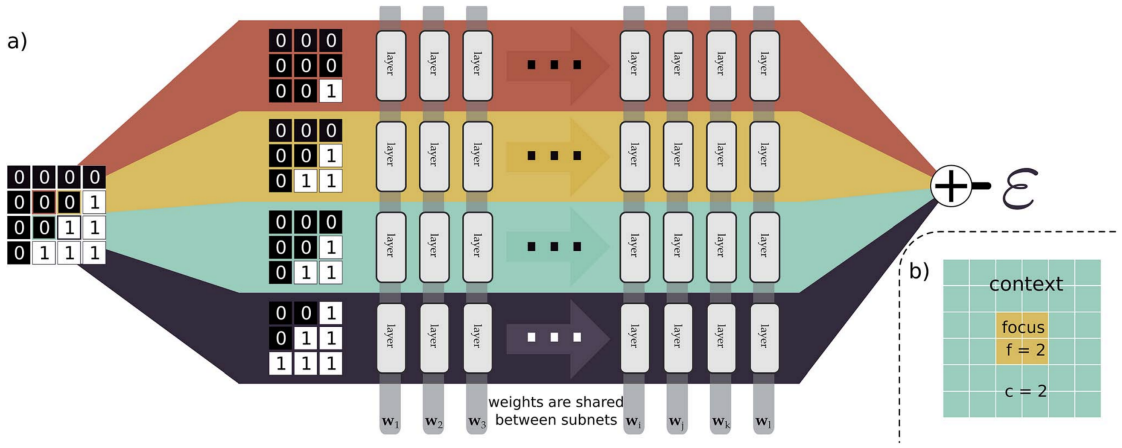


Figure 5.4: a) An example of an extensive deep neural network for a binary image with a focus of 1 and a context of 1. The image segments are fed into a network that predicts the contribution to the desired quantity. The final quantity is found by summing over the contributions. b) An example of how an image would be segmented with a focus and context value of 2. (This figure is taken from Ref. [63].)

desired quantity. This approach is similar to Ref. [25], where the total energy is found via a summation over atomic contributions.

EDNNs introduce two additional hyperparameters that must be optimized for the particular system of interest. These parameters are called *focus* and *context*, as shown in Figure 5.4b. The focus and context determine the sub-image sizes, but the context determines the amount of overlap between neighbouring sub-images. For quantities that depend on nearest neighbours, both focus and context can be set to 1. For other quantities that are non-local, however, the focus and context must be adjusted to respect the underlying interaction length scales.

5.2.5 Voxel Deep Neural Networks

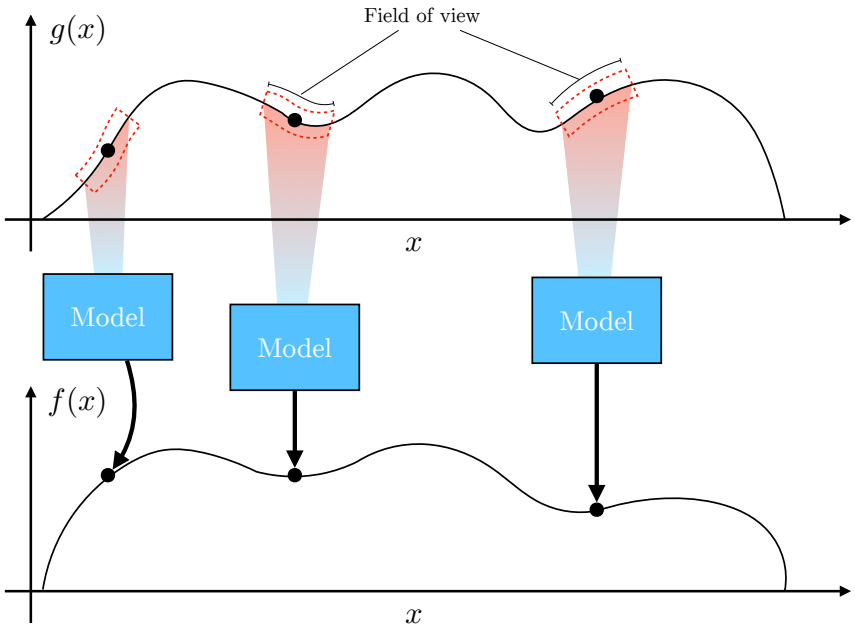


Figure 5.5: Example of a voxel deep neural network operating on a 1D function. The field of view determines the amount information to include as input to the machine learning model. We scan over the function $g(x)$, and predict the function $f(x)$. The model could be any machine learning architecture. (This is taken from a Figure of Chapter 11.)

EDNNs allow one to train on a smaller system and scale to an arbitrarily large one. However, one still must have many training examples (scalar or vector values) to construct an accurate model. Voxel deep neural networks (VDNNs) are similar to EDNNs in the sense that a function is decomposed into sub-functions, given a particular field of view, and those sub-functions are fed into a machine learning model. This can be seen in Figure 5.5, where *slices* of the function $g(x)$ are fed into a machine learning model yielding the function $f(x)$. VDNNs can be thought of as having a focus of 1, and the context defines the field of view. However, VDNNs predict scalar functions rather an scalar (or vector) values, allowing one to utilize much more information for training and requires less computation when creating a training set.

Consider a 1 dimensional system where one can define an energy density $\mathcal{E}(x)$ such that the

total energy can be found via integration

$$E = \int dx\, \mathcal{E}(x). \tag{5.26}$$

From one computation of the energy, we obtain N_x values of the energy density, where N_x is the number of grid points used to define the coordinate x. On the contrary, if we only consider the energy E, then we obtain only one data point. VDNNs allow one to construct a dataset from very few calculations if one has access to the scalar fields of interest. In addition, VDNNs also allow one to scale to an arbitrary system size. It should be noted that the model must be called thousands, if not millions of times for inference if using a real space grid in 3D. In addition, errors accumulate with integration and post processing must be done to achieve more accurate integrals.

5.3 Other Machine Learning Techniques

In this section, we discuss other machine learning techniques that have been used in the literature. As discussed in Section 1.2, two commonly used techniques include ridge regression and graph neural network (GNN) architectures. We follow Ref. [64] to describe ridge regression. In ridge regression, we have a set of structures (i.e. atomic structures, charge densities, etc.) and a set of M outputs $\{y_i\}$, where M is the number of structures. The set of structures are then mapped to a feature vector by using a *descriptor*. Common descriptors are Coulomb matrices [13], symmetry functions [11], or smooth overlap of atomic positions [15], to name a few. These feature vectors then form the set of M inputs $\{\boldsymbol{x}_i\}$. The set of outputs can be approximated by considering a function

$$y \approx f(\boldsymbol{x}) = \sum_{i=1}^{M} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i) \tag{5.27}$$

where α_i are coefficients and K is a symmetric, positive definite function that measures the similarity between input vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. This function is called a *kernel*. A commonly used kernel is a Gaussian

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\gamma |\boldsymbol{x}_i - \boldsymbol{x}_j|^2}, \tag{5.28}$$

where γ is a hyperparameter. The kernel matrix $\boldsymbol{K}$ is constructed by considering all examples in the training set, and coefficients can be found via

$$\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1} \boldsymbol{y}, \tag{5.29}$$

where λ is another hyperparameter that is used to prevent overfitting.

To discuss GNNs, we follow Ref. [65]. In GNNs, one operates on a graph that contains nodes and edges. In a chemical environment, the nodes are atoms and edges describe interactions between neighbours (or next nearest neighbours). When one considers the application of a convolutional kernel, nearest neighbour information is used to calculate an output for a given location on an image. Each pixel can be considered a node, and edges are between a given node and its nearest neighbours (or next-nearest depending on the size of the convolutional kernel). Similarly, in a graph convolution operation, information is taken from the nearest neighbors to compute a result for a particular node. However, the operation is different. In Refs. [25, 23, 21], they define an embedding for each atom type in the system. This embedding is key-value mapping, where one passes an atomic number, and obtains a vector $\boldsymbol{u}^{(n)}$. When performing a graph convolution, we obtain a new vector $\boldsymbol{u}^{(n+1)}$ via

$$\boldsymbol{u}^{(n+1)} = \sigma \left(\sum_j \boldsymbol{W}^{(n)} \boldsymbol{u}_j^{(n)} \right), \tag{5.30}$$

where $\boldsymbol{W}$ is a weight matrix with learnable parameters, σ is an activation function, and the summation runs over nearest neighbours (or all edges). This operation is performed for all nodes to achieve new embeddings. Embeddings can be updated several times, depending on how many graph convolutional layers are included within the network. To obtain a final answer (i.e. energy, forces, etc.) we can perform a summation over all of the atomic embeddings via

$$E = \sum_i \boldsymbol{W} \circ \boldsymbol{u}_i, \tag{5.31}$$

where $\boldsymbol{w}$ is a weight matrix with learnable parameters, and $\circ$ represents element-wise multiplication.

Chapter 6

Electron Transport with Tight Binding

In this Chapter we are interested in computing the current through semiconductors. We follow Refs. [66, 67] throughout. We are interested in conductors with length scales on the order of 100-1000 nm. This regime is in between microscopic and macroscopic and is called mesoscopic. Here, we assume that the mean free path of an electron is greater than the length scale of the device. The electron travels through the lattice without scattering. This is called ballistic transport. The current is written as

$$I = \frac{e}{\pi\hbar}\int_{-\infty}^{\infty} dE\, T(E)[f(E,\mu_L) - f(E,\mu_R)] \tag{6.1}$$

where f is the Fermi-Dirac distribution and the integration is done over energy. The chemical potentials, μ_L and μ_R (for left and right hand side) differ at opposing ends of the device, which drives the electrons to transmit from a higher chemical potential to a lower one. Finally, we have the transmission function $T(E)$, which is the product of the number of modes and the transmission probability for each mode. This is the central quantity that we must compute to obtain the current through a device and we focus on its computation in the following sections.

There are two approaches that are taken to solve for transmission coefficients: The Green's function approach and the wavefunction matching approach. Here, we discuss the latter. This methodology is used in Chapter 12 to calculate electron currents for graphene-based devices.

6.1 One dimensional transport through a barrier

We first discuss a one dimensional problem of an electron transmitting through a barrier as shown in Figure 6.1. For this problem, the Hamiltonian, in atomic units, is

$$H = -\frac{1}{2}\frac{d^2}{dx^2} + V(x), \tag{6.2}$$

where

$$V(x) = \begin{cases} 0 & \text{if } x < 0, \\ 1 & \text{if } 0 \leq x \leq 1, \\ 0 & \text{if } x > 1. \end{cases} \tag{6.3}$$

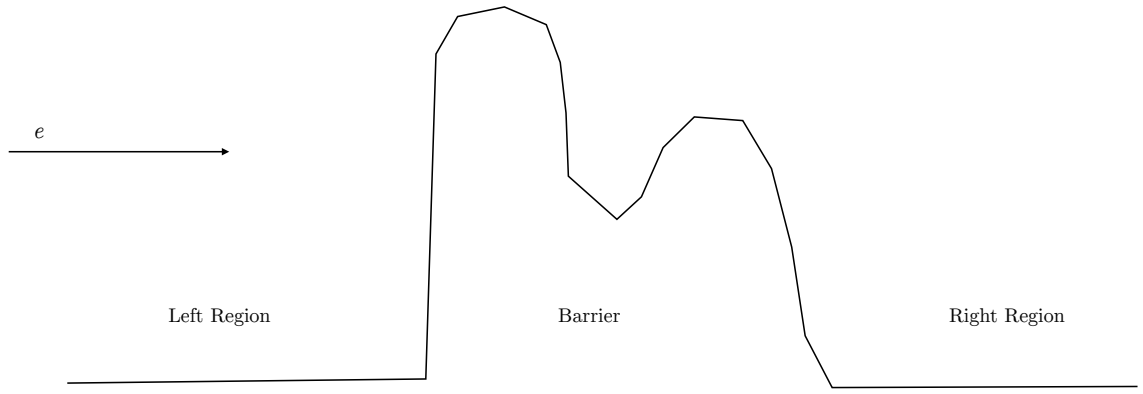


Figure 6.1: 1D transmission problem of an electron through a barrier. The wavefunction is divided into three sections, and unknowns are found through matching the wavefunction (and derivatives) at the boundaries of these sections.

We now want to solve for the transmission coefficient of this system given that on the left and right hand sides we have plane waves

$$\psi_L(x) = Ae^{ikx} + Be^{-ikx}, \quad \psi_R(x) = Fe^{ikx}, \tag{6.4}$$

where $k = \sqrt{2E}$. We only have one plane wave on the right (moving to the right) because we assume that the wavefunction is originating from the left hand side. For the wave function within the barrier, we chose to represent it on a grid. To do this we discretize our wavefunction $\psi_M(x) \equiv \{\psi_0, \psi_1, \ldots, \psi_n\}$ and potential $V(x) \equiv \{V_0, V_1, \ldots, V_n\}$ in our one dimensional space $x = \{0, a, 2a, \ldots, 1\}$ where $a = 1/(N-1)$ and N is the number of points

we have decided to discretize on. When we do this the first derivative is

$$\frac{d}{dx}\psi \equiv \frac{\psi_{n+1} - \psi_n}{a}, \tag{6.5}$$

and the second derivative is

$$\begin{aligned} \frac{d^2}{dx^2}\psi &\equiv \frac{\frac{\psi_{n+2}-\psi_{n+1}}{a} - \frac{\psi_{n+1}-\psi_n}{a}}{a} \\ &\equiv \frac{1}{a^2}(\psi_{m+1} - 2\psi_m + \psi_{m-1}), \end{aligned} \tag{6.6}$$

after making the substitution $m = n-1$. We then find that the time independent Schrödinger equation in the middle region is

$$-t\psi_{n-1} + (h_n - E)\psi_n - t\psi_{n+1} = 0, \tag{6.7}$$

where $t = 1/2a^2$, and $h_n = 2t + V_n$. We now look at the boundary conditions. At $x = 0$ we have

$$\psi_{-1} = Ae^{-ika} + Be^{ika}, \tag{6.8}$$

$$\psi_0 = A + B, \tag{6.9}$$

such that

$$\psi_{-1} = Ae^{-ika} + (\psi_0 - A)e^{ika}. \tag{6.10}$$

Additionally, at $x = 1$ we have

$$\psi_{N+1} = Fe^{ik(N+1)a} \tag{6.11}$$

$$\psi_{N+1} = \psi_N e^{ika}. \tag{6.12}$$

Our Schrödinger equation now has three pieces:

$$\begin{aligned} -t\left(Ae^{-ika}+(\psi_0-A)e^{ika}\right)+(h_0-E)\psi_0-t\psi_1 &= 0 \text{ for the left hand side,} \\ -t\psi_{n-1}+(h_n-E)\psi_n-t\psi_{n+1} &= 0 \text{ for } n\neq 0\neq N, \\ -t\psi_{N-1}+(h_N-E)\psi_N-t\psi_N e^{ika} &= 0 \text{ for the right hand side.} \end{aligned} \tag{6.13}$$

In matrix form, we obtain

$$\begin{aligned} &\begin{pmatrix} -te^{ika}+h_0-E & -t & 0 & 0 & \dots & 0 \\ -t & h_1-E & -t & 0 & \dots & 0 \\ 0 & -t & h_2-E & -t & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & -t & -te^{ika}+h_N-E \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \vdots \\ \psi_N \end{pmatrix} \\ &= \begin{pmatrix} tA(e^{-ika}-e^{ika}) \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \end{aligned} \tag{6.14}$$

This can be solved using standard linear algebra routines. Once we have ψ_N, we can solve for the transmission coefficient

$$\begin{aligned} T &= \frac{|F|^2}{|A|^2} \\ &= \frac{|\psi_N e^{-ikNa}|^2}{|A|^2} \\ &= \frac{|\psi_N|^2}{|A|^2}. \end{aligned} \tag{6.15}$$

6.2 One dimensional transport through a lattice with tight binding

In tight binding we propose an ansatz for a wavefunction that is made of atomic orbitals. This is referred to as a *linear combination of atomic orbitals* (LCAO). For electrons that are tightly bound to an atom, this is a good approximation. One could also check the bandstructure calculated with the LCAO basis versus a bandstructure calculated with *ab initio* methods to verify that the LCAO basis is sufficient. Without a loss of generality, let us assume that at each lattice site i we have one orbital $\phi(x - x_i)$. The total wavefunction of our system is then

$$\psi(x) = \sum_{i=1}^{N} c_i |\phi(x - x_i)\rangle. \tag{6.16}$$

Again, in atomic units a general one-electron Hamiltonian is

$$H = -\frac{1}{2}\frac{d^2}{dx^2} + V(x). \tag{6.17}$$

To solve for the coefficients c_i, we go back to the time independent Schrödinger equation

$$H\left(\sum_{i=1}^{N} c_i |\phi(x - x_i)\rangle\right) - E\left(\sum_{i=1}^{N} c_i |\phi(x - x_i)\rangle\right) = 0, \tag{6.18}$$

and multiply by $\langle\phi(x - x_j)|$ which yields the matrix equation

$$(\mathbf{H} - \mathbf{S}E)\mathbf{c} = 0. \tag{6.19}$$

In the above equation, the matrix $\mathbf{H}$ is called the Hamiltonian matrix and the matrix $\mathbf{S}$ is called the overlap matrix. If our basis functions ϕ_i are orthonormal and only nearest neighbour hopping is allowed, i.e.

$$\langle\psi(x - x_j)|H|\psi(x - x_i)\rangle = h \quad \text{if} \quad i = j, \tag{6.20}$$

$$\langle\psi(x - x_j)|H|\psi(x - x_i)\rangle = -t \quad \text{if} \quad i = j \pm 1, \tag{6.21}$$

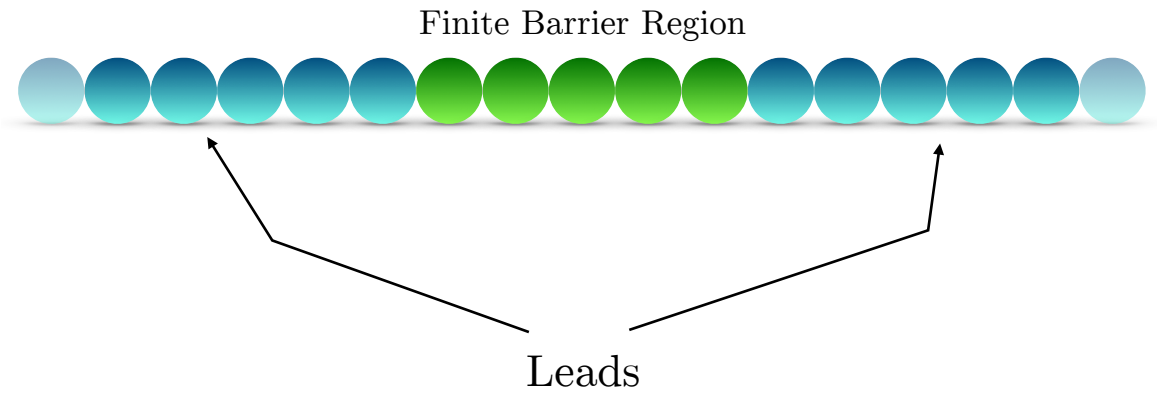


Figure 6.2: Schematic of 1D transport when using tight binding. The infinite leads are attached on either side and the finite barrier region is in the middle. We want to calculate the electron transmission coefficient after passing through the finite barrier.

then we end up with a matrix equation that looks very similar to the previous section. This matrix equation is

$$\begin{pmatrix} h-E & -t & 0 & 0 & \dots & 0 \\ -t & h-E & -t & 0 & \dots & 0 \\ 0 & -t & h-E & -t & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & -t & h-E \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} = 0. \tag{6.22}$$

For each row we have

$$-tc_{n-1} + (h-E)c_n - tc_{n+1} = 0. \tag{6.23}$$

To solve for the transmission coefficient in this system, we must introduce boundary conditions. To do this we consider semi-infinite lattices called *leads* that are attached on either end of our finite lattice. This can be seen in Figure 6.2. In the leads, we solve the above equation by using Bloch's theorem. We write

$$c_{n-1} \equiv c, \quad c_n = c\lambda, \quad c_{n+1} = \lambda^2 c. \tag{6.24}$$

This allows us to solve for λ through the quadratic equation

$$\begin{aligned} 0 &= \lambda^2 + \frac{(E-h)}{t}\lambda + 1 \qquad (6.25) \\ \lambda &= \frac{h-E}{2t} \pm \frac{\sqrt{\frac{(E-h)^2}{t^2} - 1}}{2} \\ \lambda &= \frac{h-E}{2t} \pm \sqrt{\frac{(h-E)^2}{4t^2} - 1}. \qquad (6.26) \end{aligned}$$

From here we can see that for the case

$$\frac{h-E}{2t} \leq \pm 1, \qquad (6.27)$$

we can define the wave number k such that

$$\cos(ka) = \frac{h-E}{2t}, \qquad (6.28)$$

and

$$\begin{aligned} \lambda &= \cos(ka) \pm \sqrt{\cos^2(ka) - 1} \\ &= \cos(ka) \pm i\sin(ka) \\ &= e^{\pm ika}. \qquad (6.29) \end{aligned}$$

These are the same plane wave solutions we found for a finite barrier. In addition, for the case

$$\frac{h-E}{2t} > \pm 1, \qquad (6.30)$$

we can define κ such that

$$\cosh(\kappa a) = \frac{h-E}{2t}, \qquad (6.31)$$

and

$$\begin{aligned} \lambda &= \cosh(ka) \pm \sqrt{\cosh^2(ka) - 1} \\ &= \cosh(ka) \pm \sinh(ka) \\ &= e^{\pm \kappa a}. \end{aligned} \tag{6.32}$$

These solutions are called evanescent states and they do not contribute to transport. They correspond to bound states in the scattering region. We therefore focus on solutions where we only have propagating modes when calculating transport properties in one dimension. In higher dimensions we find both evanescent and propagating modes at a constant energy. In our leads, our total wavefunction is a linear combination of the (propagating) modes. On the left hand side we have

$$c_{-1} = Ae^{-ika} + Be^{ika}, \tag{6.33}$$

$$c_0 = A + B, \tag{6.34}$$

thus

$$c_{-1} = Ae^{-ika} + (c_0 - A)e^{ika}. \tag{6.35}$$

On the right hand side we have

$$c_{N+1} = Fe^{ik(N+1)a}, \tag{6.36}$$

$$c_N = Fe^{ikNa}, \tag{6.37}$$

thus

$$c_{N+1} = c_N e^{ika}. \tag{6.38}$$

Again, we find that our Schrödinger equation has three pieces:

$$\begin{aligned} -t\left(Ae^{-ika}+(c_0-A)e^{ika}\right)+hc_0-tc_1 &= 0 \text{ for the left hand side,} \\ -tc_{n-1}+h_nc_n-tc_{n+1} &= 0 \text{ for } n\neq 0\neq N, \\ -tc_{N-1}+h_Nc_N-tc_Ne^{ika} &= 0 \text{ for the right hand side.} \end{aligned} \tag{6.39}$$

In matrix form, we obtain

$$\begin{pmatrix} -te^{ika}+h-E & -t & 0 & 0 & \dots & 0 \\ -t & h-E & -t & 0 & \dots & 0 \\ 0 & -t & h-E & -t & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & -t & -te^{ika}+h-E \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} tA(e^{-ika}-e^{ika}) \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{6.40}$$

The last thing we need to do is to find the transmission coefficient, which can be done by using the same formula as the previous section:

$$T=\frac{|c_N|^2}{|A|^2}. \tag{6.41}$$

This quantity, however, depends on energy. One sets the energy before solving for the transmission coefficient. To obtain the current, we define a set of energies that we integrate over. A similar procedure can be done in higher dimensions, where one must deal with many modes in the leads. The number of modes is proportional to the width of the lead. This is explained in more detail in Section 6.3.

6.3 Multidimensional transport through a lattice with tight binding

It is straightforward to go from one to many dimensions in tight binding. We consider the two dimensional case. The mathematics are similar for the three dimensional case. Instead

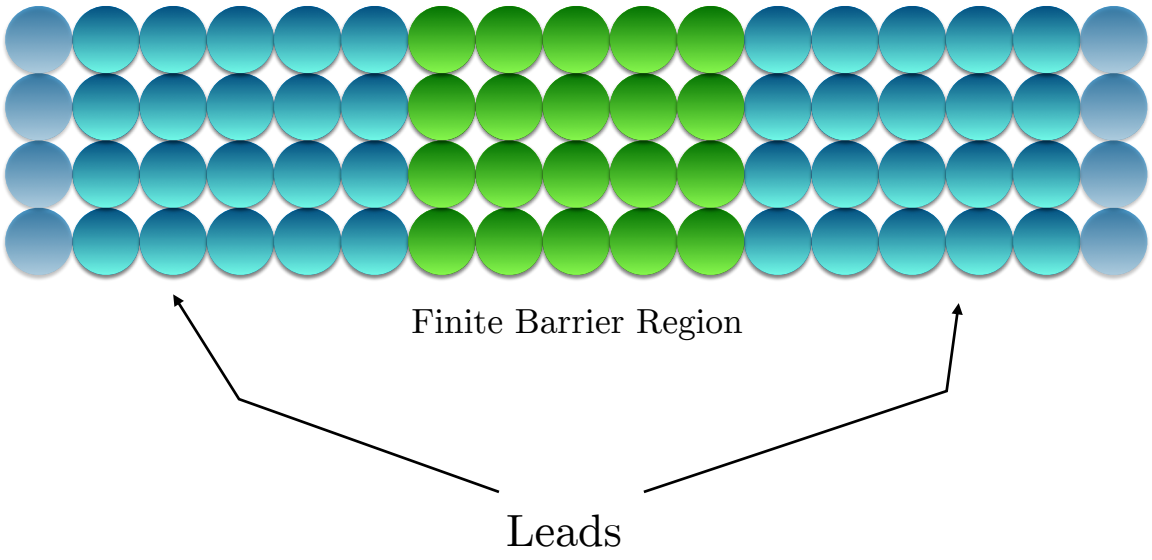


Figure 6.3: A schematic of 2D transport when using tight binding. The infinite leads are attached on either side and the finite barrier region is in the middle. We want to calculate the electron transmission coefficient after passing through the finite barrier.

of considering a row of atoms, we now consider a row of columns of atoms as shown in Figure 6.3. The wavefunction in each column takes a similar form as the one dimensional case. For each column j we have

$$\boldsymbol{\psi}_j = \boldsymbol{c}_j |\phi(\boldsymbol{x}_j)\rangle, \tag{6.42}$$

and with the same Hamiltonian and assumptions as the one dimensional case we obtain the matrix equation

$$\begin{pmatrix} \boldsymbol{H_0} - E\boldsymbol{I} & \boldsymbol{T_0} & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{T_0^\dagger} & \boldsymbol{H_1} - E\boldsymbol{I} & \boldsymbol{T_1} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{T_1^\dagger} & \boldsymbol{H_2} - E\boldsymbol{I} & \boldsymbol{T_2} & \dots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{T_{N-1}^\dagger} & \boldsymbol{H_N} - E\boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{c}_0 \\ \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \\ \vdots \\ \boldsymbol{c}_N \end{pmatrix} = \boldsymbol{0}, \tag{6.43}$$

where

$$\boldsymbol{H_n} = \begin{pmatrix} h-E & -t & 0 & 0 & \dots & 0 \\ -t & h-E & -t & 0 & \dots & 0 \\ 0 & -t & h-E & -t & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & -t & h-E \end{pmatrix}, \tag{6.44}$$

and

$$\boldsymbol{T} = \begin{pmatrix} -t & 0 & 0 & 0 & \dots & 0 \\ 0 & -t & 0 & 0 & \dots & 0 \\ 0 & 0 & -t & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & 0 & -t \end{pmatrix}. \tag{6.45}$$

We now introduce the same boundary condition as before by tacking on semi-infinite leads. In the leads the Schrödinger equation reads

$$\boldsymbol{T_{n-1}^{\dagger} c_{n-1}} + (\boldsymbol{H_n} - E\boldsymbol{I})\boldsymbol{c_n} + \boldsymbol{T}_n \boldsymbol{c_{n+1}} = 0, \tag{6.46}$$

where we again apply Bloch's theorem,

$$\boldsymbol{c_{n-1}} \equiv \boldsymbol{c}, \quad \boldsymbol{c_n} = \boldsymbol{c}\lambda, \quad \boldsymbol{c_{n+1}} = \boldsymbol{c}\lambda^2, \tag{6.47}$$

yielding the quadratic eigenvalue equation

$$\boldsymbol{T_{n-1}^{\dagger}}\boldsymbol{c} + (\boldsymbol{H_n} - E\boldsymbol{I})\boldsymbol{c}\lambda + \boldsymbol{T}_n \boldsymbol{c}\lambda^2 = 0. \tag{6.48}$$

To solve this, one can linearize it by defining $\boldsymbol{\zeta} = \boldsymbol{c}\lambda$ yielding the equation

$$\begin{pmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{T_{n-1}^{\dagger}} & (\boldsymbol{H_n} - E\boldsymbol{I}) \end{pmatrix} + \lambda \begin{pmatrix} -\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{T}_n \end{pmatrix} \begin{pmatrix} \boldsymbol{c} \\ \boldsymbol{\zeta} \end{pmatrix} = \boldsymbol{0}. \tag{6.49}$$

This equation yields $2N$ modes where N is the length of the lattice in the y direction. N modes are right moving, and the other N modes are left moving. Both real and complex

eigenvalues λ_k exist ($1 \leq k \leq 2N$) for a given value of E, which means we have both propagating and evanescent modes simultaneously. Since we have a generalized eigenvalue problem, the eigenvectors $\boldsymbol{v_k}$ are not orthogonal. To define a projection operator, one then must define dual vectors $\tilde{\boldsymbol{v}}_{\boldsymbol{k}}$ such that

$$\boldsymbol{v}_{\boldsymbol{k}}^{\dagger}\tilde{\boldsymbol{v}}_{\boldsymbol{l}} = \tilde{\boldsymbol{v}}_{\boldsymbol{k}}^{\dagger}\boldsymbol{v}_{\boldsymbol{l}} = \delta_{k,l}. \tag{6.50}$$

On the left hand side of the central region we write the lead wavefunction as a linear combination of the modes

$$\boldsymbol{c_{-1}} = \sum_{i=1}^{2N} \alpha_i \boldsymbol{v_k}, \tag{6.51}$$

or we can split up the right moving (labelled by $\rightarrow$) and left moving (labelled by $\leftarrow$) modes by writing

$$\boldsymbol{c_{-1}} = \sum_{k,\rightarrow=1}^{N} \alpha_k \boldsymbol{v_k} + \sum_{k,\leftarrow=1}^{N} r_k \boldsymbol{v_k} = \boldsymbol{c_{-1,\rightarrow}} + \boldsymbol{c_{-1,\leftarrow}}. \tag{6.52}$$

We can then project this lead wavefunction onto the dual vectors to find the coefficients, i.e.

$$\alpha_k = \tilde{\boldsymbol{v}}_{\boldsymbol{k}}^{\dagger}\boldsymbol{c_{-1,\rightarrow}}, \quad r_k = \tilde{\boldsymbol{v}}_{\boldsymbol{k}}^{\dagger}\boldsymbol{c_{-1,\leftarrow}}. \tag{6.53}$$

In addition, in the direction of periodicity we can write

$$\begin{aligned} \boldsymbol{c_0} &= \sum_{k,\rightarrow=1}^{N} \alpha_k \lambda_k \boldsymbol{v_k} + \sum_{k,\leftarrow=1}^{N} r_k \lambda_k \boldsymbol{v_k} \\ &= \sum_{k,\rightarrow=1}^{N} \lambda_k \boldsymbol{v_k} \tilde{\boldsymbol{v}}_{\boldsymbol{k}}^{\dagger} \boldsymbol{c_{-1,\rightarrow}} + \sum_{k,\leftarrow=1}^{N} \lambda_k \boldsymbol{v_k} \tilde{\boldsymbol{v}}_{\boldsymbol{k}}^{\dagger} \boldsymbol{c_{-1,\leftarrow}} \\ &= \boldsymbol{\Gamma}_{\rightarrow} \boldsymbol{c_{-1,\rightarrow}} + \boldsymbol{\Gamma}_{\leftarrow} \boldsymbol{c_{-1,\leftarrow}}. \end{aligned} \tag{6.54}$$

In the leads, one can apply the Bloch matrix

$$\boldsymbol{\Gamma}^n = \sum_{k=1}^{N} \lambda_k^n \boldsymbol{v_k} \tilde{\boldsymbol{v}}_{\boldsymbol{k}}^{\dagger}, \tag{6.55}$$

where n is a positive or negative integer that takes you from column i to column $i+n$. Using

the Bloch matrices, we write on the left hand side of the central region

$$\begin{aligned} \boldsymbol{c_0} &= \boldsymbol{c_{0,\rightarrow}} + \boldsymbol{c_{0,\leftarrow}} \\ \boldsymbol{c_{-1}} &= \boldsymbol{\Gamma_{\rightarrow}^{-1}} \boldsymbol{c_{0,\rightarrow}} + \boldsymbol{\Gamma_{\leftarrow}^{-1}} \boldsymbol{c_{0,\leftarrow}} \\ \boldsymbol{c_{-1}} &= (\boldsymbol{\Gamma_{\rightarrow}^{-1}} - \boldsymbol{\Gamma_{\leftarrow}^{-1}}) \boldsymbol{c_{0,\rightarrow}} + \boldsymbol{\Gamma_{\leftarrow}^{-1}} \boldsymbol{c_0}. \end{aligned} \tag{6.56}$$

Using the same procedure on the right hand side of the central region we have

$$\boldsymbol{c_{N+1}} = \boldsymbol{\Gamma_{\rightarrow}} \boldsymbol{c_N}, \tag{6.57}$$

since we don't have any left moving modes. Putting everything together, we find

$$\begin{aligned} [(\boldsymbol{\Gamma_{\rightarrow}^{-1}} - \boldsymbol{\Gamma_{\leftarrow}^{-1}}) \boldsymbol{c_{0,\rightarrow}} + \boldsymbol{\Gamma_{\leftarrow}^{-1}}] \boldsymbol{T_{-1}^{\dagger}} \boldsymbol{c_0} + (\boldsymbol{H_0} - E\boldsymbol{I}) \boldsymbol{c_0} + \boldsymbol{T_0} \boldsymbol{c_1} &= 0 \text{ on the left hand side,} \\ \boldsymbol{T_{n-1}^{\dagger}} \boldsymbol{c_{n-1}} + (\boldsymbol{H_n} - E\boldsymbol{I}) \boldsymbol{c_n} + \boldsymbol{T_n} \boldsymbol{c_{n+1}} &= 0 \text{ in the middle,} \\ \boldsymbol{T_{N-1}^{\dagger}} \boldsymbol{c_{N-1}} + (\boldsymbol{H_N} - E\boldsymbol{I}) \boldsymbol{c_N} + \boldsymbol{\Gamma_{\rightarrow}} \boldsymbol{T_N} \boldsymbol{c_N} &= 0 \text{ on the right hand side,} \end{aligned} \tag{6.58}$$

or in matrix form

$$\begin{pmatrix} \boldsymbol{\Gamma_{\leftarrow}^{-1}} \boldsymbol{T_{-1}^{\dagger}} + \boldsymbol{H_0} - E\boldsymbol{I} & \boldsymbol{T_0} & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{T_0^{\dagger}} & \boldsymbol{H_1} - E\boldsymbol{I} & \boldsymbol{T_1} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{T_1^{\dagger}} & \boldsymbol{H_2} - E\boldsymbol{I} & \boldsymbol{T_2} & \dots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{T_{N-1}^{\dagger}} & \boldsymbol{H_N} - E\boldsymbol{I} + \boldsymbol{\Gamma_{\rightarrow}} \boldsymbol{T_N} \end{pmatrix} \times \begin{pmatrix} \boldsymbol{c_0} \\ \boldsymbol{c_1} \\ \boldsymbol{c_2} \\ \vdots \\ \boldsymbol{c_N} \end{pmatrix} = \begin{pmatrix} (\boldsymbol{\Gamma_{\rightarrow}^{-1}} - \boldsymbol{\Gamma_{\leftarrow}^{-1}}) \boldsymbol{T_{-1}^{\dagger}} \boldsymbol{c_{0,\rightarrow}} \\ \boldsymbol{0} \\ \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \end{pmatrix}. \tag{6.59}$$

The $\boldsymbol{c}_{0,\rightarrow}$ in the source vector is replaced by each right moving mode. For each mode, one solves the set of linear equations. Once the coefficients have been solved for in the central region, we can find the transmission coefficient by projecting the coefficients of the last layer onto the modes in the right lead. Mathematically speaking, we write

$$\boldsymbol{c}_{\boldsymbol{N}} = \sum_{k,\rightarrow=1}^{N} t_k \boldsymbol{v}_{\boldsymbol{k}}, \tag{6.60}$$

such that the transmission coefficients are

$$t_{\ell,k} = \tilde{\boldsymbol{v}}_{\boldsymbol{\ell}}^{\dagger} \boldsymbol{c}_{\boldsymbol{N}}. \tag{6.61}$$

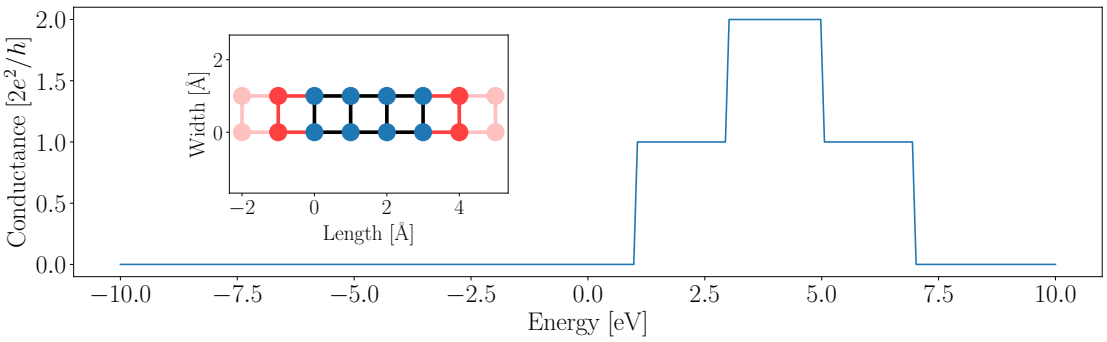


Figure 6.4: Conductance (transmission function in conventional units) versus energy for a rectangular lattice with a width of two. In the inset, we show the structure that we calculate the conductance for. The red nodes represent leads and the blue nodes represent the finite region.

Once we have computed all of the transmission coefficients between the ℓ right moving modes from the left lead and the k right moving modes from the right lead we can write the transmission function as

$$T = \sum_{\ell,\rightarrow=0}^{N} \sum_{k,\rightarrow=0}^{M} |t_{\ell,k}|^2. \tag{6.62}$$

In Figure 6.4, we show a 2D regular lattice structure and the conductance versus energy. Here, the hopping value is set to $t = -1$ eV and the on-site energy is set to $h = 4$ eV. The conductance is the transmission function multiplied by $2e^2/h$. In this plot we see jumps in

the conductance as the energy is increased. These jumps correspond with the modes from the leads being activated at a given energy. Since the width of the structure is 2, we obtain a maximum conductance of 2. This is because the maximum number of modes that contribute to transport is 2.

Part III

Articles

Chapter 7

Convolutional Neural Networks for Atomistic Systems

In this Chapter, we present our first article on accelerating the computation of a nanoscale material. To do so, we use deep convolutional neural networks (CNNs) to compute properties of simple diatomic molecules and hexagonal sheets with an image-based representation. Previously, deep learning had been used in computer vision tasks without any feature engineering, which gave rise to the term *featureless* learning. Features are extracted during the training process such that a local optima is found. Using this methodology, we are able to predict energies for hexagonal sheets on the order of milliseconds while maintaining chemical accuracy. In the future, such a model can be used in a materials design pipeline to enable the search of materials given a target property. This could be done by simply performing rapid inference and using an evolutionary strategy or by using automatic differentiation (gradients) to achieve a particular input structure given a constant output. This work was published in 2018 in Computational Materials Science.

Kevin Ryczko,
Department of Physics, University of Ottawa

Kyle Mills,
Department of Physics, University of Ontario Institute of Technology

Iryna Luchak,
Department of Electrical & Computer Engineering, University of British Columbia

Christa Homenick,
National Research Council of Canada

Isaac Tamblyn,
National Research Council of Canada,
Department of Physics, University of Ottawa

Abstract

We introduce a new method, called CNNAS (convolutional neural networks for atomistic systems), for calculating the total energy of atomic systems which rivals the computational cost of empirical potentials while maintaining the accuracy of *ab initio* calculations. This method uses deep convolutional neural networks (CNNs), where the input to these networks are simple representations of the atomic structure. We use this approach to predict energies obtained using density functional theory (DFT) for 2D hexagonal lattices of various types. Using a dataset consisting of graphene, hexagonal boron nitride (hBN), and graphene-hBN heterostructures, with and without defects, we trained a deep CNN that is capable of predicting DFT energies to an extremely high accuracy, with a mean absolute error (MAE) of 0.198 meV / atom (maximum absolute error of 16.1 meV / atom). To explore our new methodology, we investigate the ability of a deep

neural network (DNN) in predicting a Lennard-Jones energy and separation distance for a dataset of dimer molecules in both two and three dimensions. In addition, we systematically investigate the flexibility of the deep learning models by performing interpolation and extrapolation tests.

Keywords: Dimer molecules, deep learning, convolutional neural networks, density functional theory, 2D materials

7.1 Introduction

Solving the electronic structure problem has long been of interest to researchers in materials science, chemistry, and physics. Even modest systems consisting of a few atoms are impossible to treat exactly and simplifications or approximations must be made to reduce the complexity of the problem. This could be in the form of the Born-Oppenheimer approximation [68], frequently used in conjunction with Kohn-Sham DFT [69], or the use of phenomenological fits to a set of experimental or theoretical results. Although approximate electronic structure methods have the advantage that they preserve the characteristics of the underlying physics (e.g. the wavefunction or ground state charge density is treated as a fundamental object), they are limited in applicability due to unfavourable scaling with system size and computational cost [70]. In the domain of phenomenological fits, force fields informed from high-level theory calculations and experiment have seen success [71, 72, 73, 74, 75, 76, 77] in modelling phenomena that occur on time and length scales beyond the reach of the lower level electronic structure methods. Force fields even predate the quantum theory itself; the van der Waals equation of state is dependent upon two species-specific fitting parameters argued for based on microscopic atomic interactions [78]. Interaction terms evocative of the van der Waals parameters still appear in many modern force-fields [79, 80, 81, 82].

Approximations and phenomenological fits are useful in materials discovery and design, where oftentimes a specific property (e.g. band gap, ionization energy, etc.) is desired. Targeting the search at novel materials with a specific property is a difficult task given the large search space spanned by permutations of atoms. Thus the ability to make rapid, accurate predictions about prospective materials is invaluable. Although the term "machine learning" has not traditionally been applied to phenomenological fits, the task of reproducing a generalized mapping of input-to-output through a series of observations is the core of supervised machine learning.

Many recent studies have used machine learning in some form to study molecular or condensed matter systems [16, 83, 18, 84, 85, 24, 86, 87, 88, 89, 17, 13, 90, 15]. In particular, Gómez-Bombarelli et al. [84] used auto encoders to project their training set onto a latent space. They were able to operate within this latent space, exploring chemical space (by proxy), and found new molecules with desirable properties that were not present in the original data set. The input to their auto encoder was based off of the simplified molecular input line entry system (SMILES)[91], a descriptor consisting of a minimal string of text to describe the three dimensional molecule. Other works using kernel ridge regression (KRR) [18, 83, 85, 24, 86], rely on abstract constructions of input feature vectors describing the chemical system. Rupp et al. [85] used principal component analysis (PCA) to obtain a three dimensional atom-centered local coordinate system that was then used as input into KRR. This framework was successful in predicting nuclear magnetic resonance (NMR) chemical shifts, core ionization energies, and atomic forces. Another choice of a feature vector is the one given by Behler and Parrinello [11]. This feature vector was constructed with an artificial neural network (ANN) architecture in mind, and is written in terms of symmetric functions that obey rotational and translational invariances. Preceding this descriptor, Bartók *et al.* [15] showed that a new approach, called Smooth Overlap of Atomic Positions (SOAP), eliminates the need for ad hoc descriptions of atomic environments. They showed that by directly defining the similarity between atomic environments, they could still include symmetric and invariant properties, necessary to describe atomic environments. This approach was then successfully applied to fit the potential energy surfaces of different silicon structures, and was also successfully applied in another report [92] to traverse through chemical space and make energy predictions for small molecules within chemical accuracy. Additional works include using a Coulomb matrix to make predictions of atomization energies [13], and understanding of machine learning density functionals [90].

Our alternative approach to overcome the challenge of finding a suitable input feature vector is inspired by the recent successes of applying "big data" to grand challenge problems in computer vision and computational games [93, 94]. Rather than seeking to simplify, compress, or approximate the interactions within a system, we train a highly flexible, data-driven model on a large number of "ground truth" examples. Here we argue that this

brute force approach may offer a more scalable and parallelizable approach to large-scale electronic structure problems than existing methods can offer. In our approach, we use a simple, image-based approximate representation of the electrostatic potential, that preserves spatial structure. By avoiding the construction of an input feature vector, we sidestep any possibility of introducing biases, and preserve the spatial correlations between atoms in the training examples.

While many machine learning algorithms exist, we have focused on a particular class: deep CNNs. CNNs have the ability to "learn" non-linear input-to-output mappings without prior formulation of a model or functional form. This is done through the use of convolutional layers. A convolutional layer consists of a set of kernels (matrices) which contain adjustable parameters. When an operation is performed on an image with a convolutional layer, the operation produces a different image for every kernel in that layer. The kernel moves from pixel to pixel in the image, and performs the dot product between the weights and the pixels enclosed by the size of the kernel. The result is a convolved image. While training, the parameters in the kernels are updated so that they have the ability to enhance features in the images necessary for making accurate predictions. Incorporating convolutional operations allows the neural network to exploit the spatial structure naturally present in the input data, and drastically reduces the number of redundant trainable parameters (compared to a traditional ANN). During the training procedure, the deep neural network automatically "learns" by optimizing a set of features necessary to reproduce the desired input-to-output mapping. Recently, Mills *et al.* [63] was able to reach chemical accuracy, applying a deep CNN to the one-electron Schrödinger equation in two dimensions. The input to the system was solely the external potential. During the training process, the neural network used the information in a large collection of these potentials to develop a set of features necessary to reproduce the energies. The features that a deep CNN develops are not directly interpretable, but collectively form a latent space in which the desired mapping can be interpolated. Deep neural networks excel at interpolation within the latent space, but perform poorly when extrapolating (as we show in Subsection 7.3.2). One disadvantage of our approach is the discretization of the atomic coordinates on a real space grid. This means that the convolutional kernels that are applied onto the image are

also discrete. Additionally, the images that are fed into our network must have the same dimensions as the images the model was trained on. In a recent study [95], Schütt *et al.* avoided discretization errors by using a continuous convolutional approach. This approach was successful at predicting both energies and forces for small molecules, and avoided constructing atomic fingerprint functions by optimizing the fingerprints in the training phase. Similar to previously mentioned work, information of the local environment (e.g. radial distance of neighbouring atoms) was needed to construct the convolutional kernels. In our approach, the optimal environmental features are calculated during training.

In this Article, we first describe our new method to calculate total energies of atomic systems. This includes how to construct images that are used as input to the CNNs. These images describe the atomic environment, and can be generalized to *any* atomic system. We then explore and test the limitations of our methodology for a model system. We use our method to predict distances between dimer pairs in both two and three dimensions. Additionally, for each pair we compute a Lennard-Jones energy, and demonstrate that our methodology can predict this energy to a high degree of accuracy. We perform interpolation and extrapolation tests, and vary different controllable parameters to further understand our methodology. Using this success as motivation, we then use our new methodology to predict the total energy according to DFT within the generalized gradient approximation to a very high accuracy for various hexagonal lattices: graphene, boron nitride, and graphene-boron nitride heterostructures. Our method is also able to predict energies for structures containing vacancies and Stone-Wales (five-seven) defects.

7.2 Methods

7.2.1 Input representation

Since our method uses a deep CNN which exploits spatial structure in the input data structure, we decided to represent our atomic configurations as the approximate nuclear potential evaluated on a real-space mesh. While a Coulomb potential is the initial obvious choice, we used an atom-centered Gaussian representation to avoid the diverging Coulomb

singularity. We evaluate our function on a real space grid with the value at point (x, y, z) given by:

$$V(x,y,z) = \sum_{i=1}^{N} Z_i \exp\left(-\frac{[(x-x_i)^2+(y-y_i)^2+(z-z_i)^2]}{2\gamma^2}\right) \tag{7.1}$$

where x_i, y_i, z_i are the coordinates of atom i with atomic number Z_i. We chose $\gamma = 0.2$ Å as the width of the Gaussian peaks, consistent with Brockherde *et al.* [17]. For the two dimensional images, the z coordinate is not included. We note that the choice of V is arbitrary, so long as it is consistent, and the relative peaks of the atoms are maintained.

7.2.2 The datasets

The datasets we generated consisted of two groups: dimer pairs and hexagonal sheets. For the dimer dataset, we randomly generated two position vectors, $\vec{r}_1$ and $\vec{r}_2$, so that the distance between the two points $r_{12} = |\vec{r}_1 - \vec{r}_2|$ was within a specified range of values (e.g. $1.0 \leq r_{12} \leq 2.0$ Å). To accomplish this, we place the first atom down randomly, and then place down the second atom so that the distance between the two is maintained. The angle between the two position vectors is also randomly chosen when placing the second atom. To make sure that we do not reproduce a previously generated image, we declared a minimum dimer molecule overlap distance of 0.01 Å. With this overlap distance specified, we then initialized arrays for every grid point on a discretized grid ($0 \leq x \leq 10$ Å) with a $\Delta x = 0.01$ Å spacing. When we generated positions for the first dimer molecule, we found the arrays associated with the dimer coordinates, and added the index of this molecule to these arrays. When we generated additional positions for dimer molecules we first found the arrays associated with these proposed coordinates and checked to see if there is an intersection of indices between these two arrays as well as the arrays associated with neighbouring grid points. If there was an intersection, we proposed new coordinates. If not, we recorded the index in the arrays and continued generating new positions. Using these coordinates, we evaluate Eq. (7.1) on a 256×256 grid-point mesh for 2D, and a $64 \times 64 \times 64$ grid-point mesh for 3D. In both cases, the real-space length of one side of the mesh is 10 Å. Both dimer atoms have the same atomic number: $Z_1 = Z_2 = 1$. We generated 500,000 2D images and 100,000 3D images. For each dimer image, we recorded two labels on which we would ultimately

train the deep neural network: the distance r_{12} and the Lennard-Jones energy

$$U(r_{12}) = 4\epsilon \left[\left(\frac{\sigma}{r_{12}} \right)^{12} - \left(\frac{\sigma}{r_{12}} \right)^{6} \right], \tag{7.2}$$

where we take $\sigma = 1$ Å and $\epsilon = 1$. Note that one can construct a CNN to predict both the distance and energy simultaneously.

To generate the configurations of hexagonal sheets, we performed Born-Oppenheimer molecular dynamics (BOMD) using DFT on systems of graphene, hBN, and a graphene-hBN heterostructure, all consisting of 60 atoms. Additionally, we generated datasets with single point defects, as well as Stone-Wales defects. The calculations were carried out using VASP [96, 97, 98, 99], with the PBE exchange correlation functional [44]. All of the supercell dimensions were $12.53 \times 13.02 \times 10.0$ Å, and the atoms were constrained along the z-axis at $z = 0.0$ to allow for a two-dimensional treatment. We used a Nosé-Hoover thermostat of 1000 K, a plane wave kinetic-energy cutoff of 500 eV, and a $\mathbf{k}$-point grid of $2 \times 2 \times 1$ centred about the Γ point. For the MD, a timestep of 11.3 a.u. was used in the simulations. For each type of hexagonal sheet, the training set was generated by running many independant sets of MD for 0.15 picoseconds (approximately 550 steps). After the completion of one MD run, the final coordinates were randomly translated in the x and y-directions, and the velocities were reinitialized using a Maxwell-Boltzmann distribution. This process was repeated until approximately 9 picoseconds per hexagonal structure was generated. In total, we generated 269,016 images in our generation process. Using the coordinates of the atoms from the molecular dynamics frames, the training images were generated using Eq. (7.1), summing over all atoms in the unit cell. The atomic numbers, Z_i were used so that atoms of higher atomic number had a larger Gaussian peak. The pixels were wrapped to obey periodic boundary conditions.

After the generation of all datasets, they were then split randomly so that 70% comprised a training set and 30% comprised a testing set. While training, 10% of the training set was used as a validation set. All of the testing sets were only used to compute errors, and were not be accessible to the neural network during the training process. Some example images of the input datasets are shown in Figure 7.1.

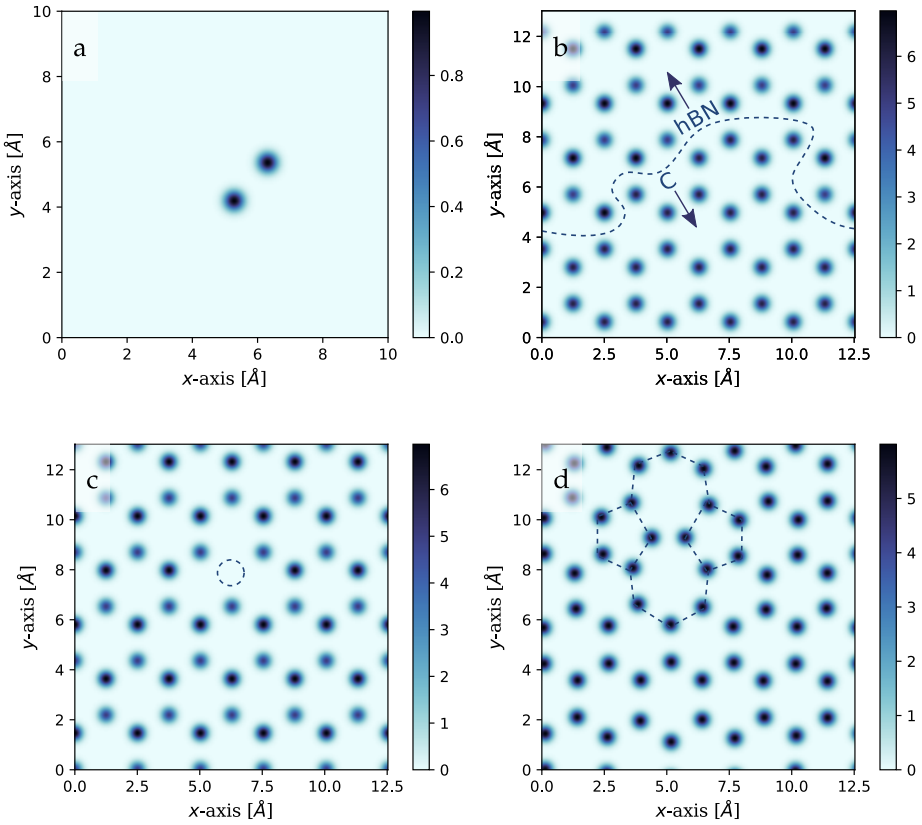


Figure 7.1: Images used as input to the CNNs. (a): A dimer molecule image. (b): An image of a graphene-hBN heterostructure. The dashed lines divides the hBN structure and graphene. (c): An image of hBN with a N atom removed (single point defect, indicated with dashed line). (d): An image of graphene with a Stone-Wales defect in the region indicated by the dashed lines.

7.2.3 The CNNs

We used two relatively deep neural network architectures, shown in Figure 7.2, comprised of a combination of reducing (stride 2) and non-reducing (stride 1) convolutional layers (CLs). CLs consist of an array of kernels that operate on images. The kernel sizes of CLs determine the number of parameters that are optimized during the training process. When using a reducing CL, the images that are output from the CL will have reduced dimensionality. When using a non-reducing CL, the images that are output from the CL will have the same dimensionality. We do not use dropout, or any sort of pooling in our network architectures. Network 1 has the identical architecture used in [63], and was used for learning the dimer

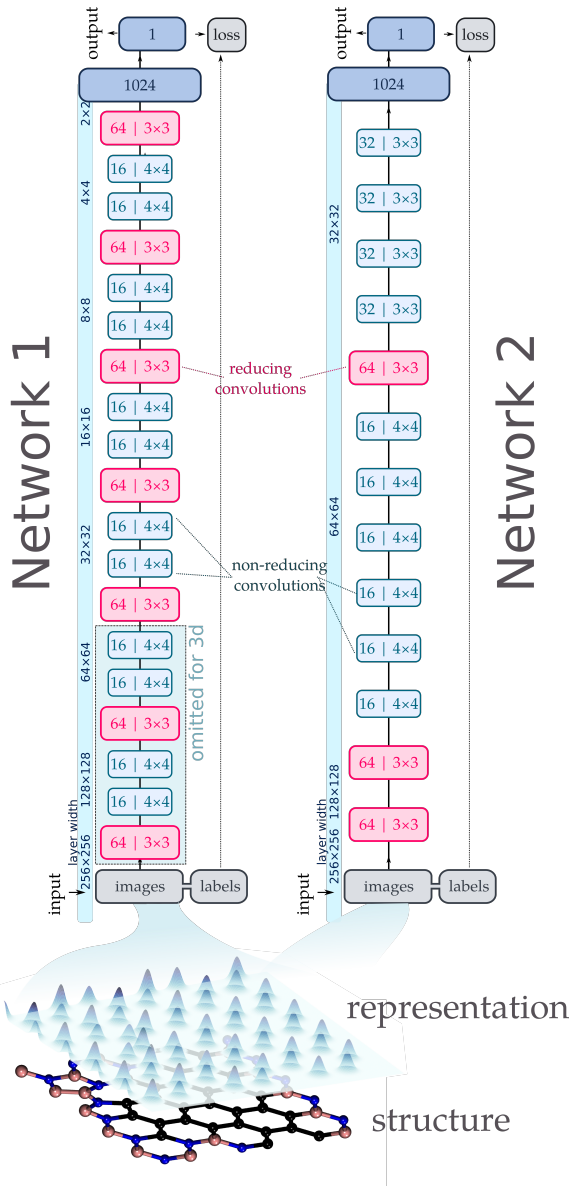


Figure 7.2: Schematic description of the methodology. The atomic structures are first mapped to a Gaussian surface described by Equation 7.1. These images are then passed into the CNNs. Network 1 was used for all dimer models, and Network 2 was used for the hexagonal structures. Reducing convolutional layers operate with a stride of 2 in each direction, and reducing convolutional layers operate with unit stride. In the diagram, the number of filters and the filter dimension (for two dimensions) are shown. These networks were optimized in parallel on graphical processing units (GPUs) using Tensorflow [100].

distances and energies. For the 3D models, three dimensional filters were used (e.g. $4 \times 4 \times 4$ instead of 4×4), and certain layers (shown in Figure 7.2) were omitted to accommodate the smaller input dimension. Within the convolutional layers, we used zero padding when

applying convolutional kernels to the edge pixels. This allows for the dimensionality of the convolved images to remain the same for non-reducing layers, and to decrease exactly by half in the reducing layers. Network 2 was modified for better performance on the hexagonal sheets. When constructing this model, our guiding principle was simplicity. We used trial and error in removing layers from Network 1 until we were able to reach a certain accuracy in our predictions. It is very likely that a more optimized architecture (found through a service like SigOpt [101]) could result in even better performance. In order to include periodic boundary conditions in our method, we used four shifted and wrapped copies of each image during training. The four copies constitute a shift such that each of the four original boundaries appear at the centre of the image in at least one of the copies. This leads to a network topology with 4 neural networks (with the same weights) being training concurrently, a fully connected layer with 1024 neurons to combine the output layers of the 4 networks, and final fully connected layer that outputs the prediction. We used rectified linear units (ReLU) for all activations, and trained using the Adam optimization scheme [60] to minimize the mean-squared error between the correct energy/distance and the CNN output. The use of ReLU activations mean that the computed gradients with respect to weights will be constant, which improves the efficiency of the backpropagation algorithm and is less demanding to evaluate than the sigmoid function (since the derivatives of the activation functions with respect to weights are constant). For the dimer models, we trained for 500 epochs with a constant learning rate of 10^{-4}. For the model making energy predictions of the hexagonal sheets we trained for 300 epochs with a learning rate of 10^{-5}, and then dropped the learning rate to 10^{-6} before training for another 200 epochs. All of the models were trained using TensorFlow [100].

7.3 Results

7.3.1 Dimer pairs

We demonstrate the ability of the CNNAS approach at predicting the separation distance between dimer pairs and the Lennard-Jones energies using the following four models:

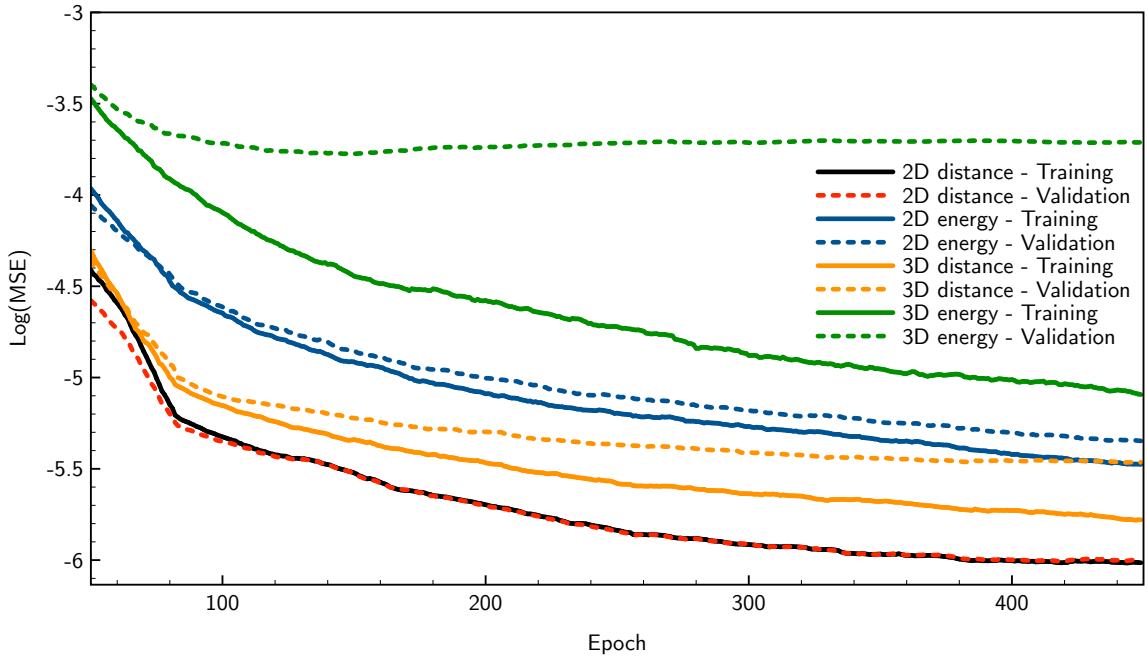


Figure 7.3: Running averages (with a window of 50) of the mean squared errors (loss) recorded during training. The models are the following: A 2D model for predicting distances in the range $1 \leq r_{12} \leq 3$ (2D distance), a 3D model for predicting distances in the range $1 \leq r \leq 3$ (3D distance), a 2D model for predicting energies in the range $-1 \leq U(r_{12}) \leq 0$ (2D energy), and a 3D model for predicting energies in the range $-1 \leq U(r_{12}) \leq 0$ (3D energy). The units of the loss function is in Å^2 for the distance models, and ϵ^2 for the energy models.

1. a 2D model for predicting distances in the range $1 \leq r_{12} \leq 3$,
2. a 3D model for predicting distances in the range $1 \leq r_{12} \leq 3$,
3. a 2D model for predicting energies in the range $-1 \leq U(r_{12}) \leq 0$, and
4. a 3D model for predicting energies in the range $-1 \leq U(r_{12}) \leq 0$.

In Figure 7.3 we plot the running average of the training and validation loss as a function of epoch (one time through the training set) for each model. After five hundred epochs, we see that the CNNs are converged (Figure 7.3). When comparing the models in Table 7.1, we find that the 2D models outperform the 3D models in all cases. This can be attributed to the amount of training data provided to both models. Since the 2D models are significantly faster to train, we were able to provide the deep neural network with more training examples. To investigate this, we trained a 3D model with 500,000 images and found that the MAE

Grid size	System description	MAE (testing set)	MSE (validation set)
256×256	$1 \leq r_{12} \leq 3$	1.58×10^{-3} Å	5.52×10^{-6} Å^2
256×256	$1 \leq r_{12} \leq 2$	9.99×10^{-4} Å	1.48×10^{-6} Å^2
256×256	$2 \leq r_{12} \leq 3$	9.72×10^{-4} Å	1.47×10^{-6} Å^2
256×256	$1 \leq r_{12} \leq 1.67$ & $2.33 \leq r_{12} \leq 3$	3.52×10^{-3} Å	3.48×10^{-6} Å^2
$64 \times 64 \times 64$	$1 \leq r_{12} \leq 3$	3.21×10^{-3} Å	1.64×10^{-5} Å^2
$64 \times 64 \times 64$	$1 \leq r_{12} \leq 2$	1.58×10^{-3} Å	4.20×10^{-6} Å^2
$64 \times 64 \times 64$	$2 \leq r_{12} \leq 3$	1.72×10^{-3} Å	4.77×10^{-6} Å^2
$64 \times 64 \times 64$	$1 \leq r_{12} \leq 1.67$ & $2.33 \leq r_{12} \leq 3$	2.93×10^{-3} Å	1.50×10^{-5} Å^2
256×256	$-1 \leq U(r_{12}) \leq 0$	1.41×10^{-3} ϵ	5.82×10^{-6} ϵ^2
256×256	$-0.5 \leq U(r_{12}) \leq 0$	1.62×10^{-3} ϵ	5.29×10^{-6} ϵ^2
256×256	$-1 \leq U(r_{12}) \leq -0.5$	1.62×10^{-3} ϵ	2.33×10^{-6} ϵ^2
256×256	$-1 \leq U(r_{12}) \leq 0.67$ & $-0.33 \leq U(r_{12}) \leq 0$	1.41×10^{-3} ϵ	3.72×10^{-6} ϵ^2
$64 \times 64 \times 64$	$-1 \leq U(r_{12}) \leq 0$	5.11×10^{-3} ϵ	1.91×10^{-4} ϵ^2
$64 \times 64 \times 64$	$-0.5 \leq U(r_{12}) \leq 0$	1.14×10^{-2} ϵ	3.19×10^{-4} ϵ^2
$64 \times 64 \times 64$	$-1 \leq U(r_{12}) \leq 0.67$ & $-0.33 \leq U(r_{12}) \leq 0$	9.43×10^{-3} ϵ	2.05×10^{-4} ϵ^2
32×32	$-1 \leq U(r_{12}) \leq 0$	4.99×10^{-1} ϵ	3.31×10^{-1} ϵ^2
64×64	$-1 \leq U(r_{12}) \leq 0$	1.70×10^{-3} ϵ	4.02×10^{-6} ϵ^2
128×128	$-1 \leq U(r_{12}) \leq 0$	1.06×10^{-3} ϵ	1.67×10^{-6} ϵ^2
256×256	$1 \leq r_{12} \leq 2$ (Random forest)	1.68×10^{-2} Å	-
256×256	$1 \leq r_{12} \leq 2$ (Kernel ridge regression - linear)	1.24×10^{-1} Å	-
256×256	$1 \leq r_{12} \leq 2$ (Kernel ridge regression - Gaussian)	2.47×10^{-1} Å	-
256×256	$1 \leq r_{12} \leq 2$ (Multilayer perceptron)	1.24×10^{-1} Å	-
256×256	Hexagonal sheets	0.0119 eV	3.11×10^{-4} eV^2

Table 7.1: Mean absolute errors and mean squared errors of various systems with their corresponding test and validation sets. The mean squared errors are taken from epoch 500.

decreases by 85%. Additionally, the difference in training data resolution plays a role. The pixel density in the 2D dimer dataset is higher than in the 3D dimer dataset. When testing a 2D energy predicting model on a 64×64 grid (2.44×10^{-2} Å^2 pixel area) rather than a 256×256 grid (1.53×10^{-3} Å^2 pixel area), the MAE dropped to 1.70×10^{-3} ϵ (21% difference). To further investigate the grid sizes, we calculated the MAEs of test sets for 2D energy models with identical dataset sizes but differing grid sizes. We changed the grid sizes from 32×32 to 128×128 in multiples of 2, and found that an energy predicting model performed best with a 128×128 grid. In addition to investigating the grid size, we also carried out experiments where we upscaled lower dimensional images (e.g. 64×64 grids) into 256×256 grids. This was done by replicating the pixels in the lower dimensional image. When upscaling from 64×64 to 256×256, each pixel would be replicated 16 times (4 in the horizontal direction and 4 in the vertical). We found a similar MAE when comparing the upsampled images and the original images. A non-upscaled 256×256 grid contains enough information for the DNN to make accurate predictions. A 256×256 resolution image is sufficient to generate an accurate DNN for our atomistic systems.

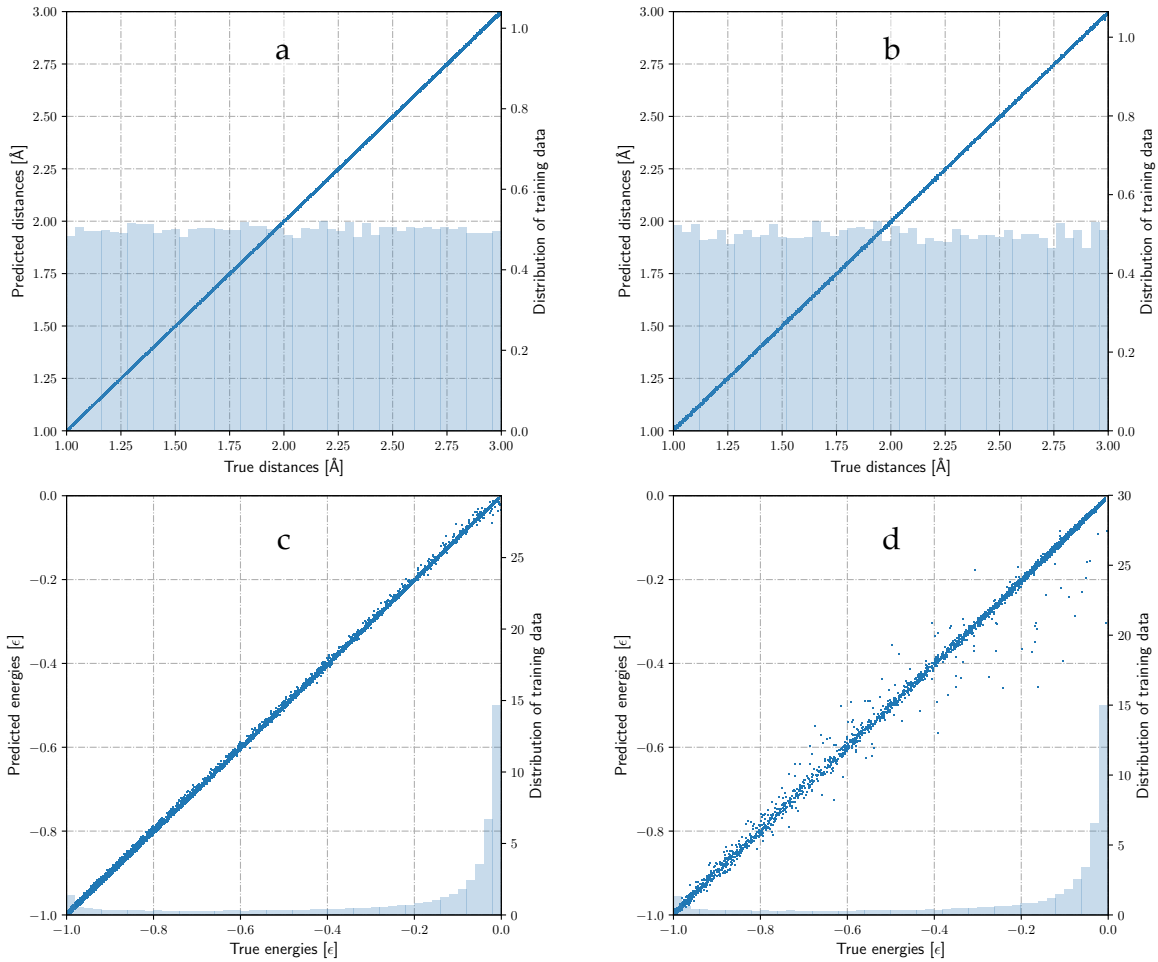


Figure 7.4: Predicted versus true plots for various models. (a) The 2D model for predicting distances in the range $1 \leq r_{12} \leq 2$. (b) The 3D model for predicting distances in the range $1 \leq r \leq 2$. (c) The 2D model for predicting energies in the range $-1 \leq U(r_{12}) \leq 0$. (d) the 3D model for predicting energies in the range $-1 \leq U(r_{12}) \leq 0.0$. The light blue distribution in the background of all plots shows the distribution of data used in the training of the models.

The differences between the different models can also be seen in Figure 7.4, where we plot the predicted versus true values for these models as well as the distribution of input data. For the 3D predicted versus true scatter plots, there is much more variance in the distribution of points in comparison to the 2D models which is due to less training data and the pixel density, as discussed above. When comparing distance models with the energy models, we can visually see that the variance in the energy distributions are higher than the distance models. To investigate this systematically, we trained 3 independent models:

1. A 2D model trained on a harmonic function $U(r) = (r - 2.0)^2$ for $1.0 \leq r \leq 3.0$.

2. A 2D model trained on the function $U(r) = \frac{1}{r^{12}}$ for $1.0 \leq r \leq 3.0$.

3. A 2D energy model with uniform sampling.

For model trained on the harmonic function and the function r^{-12}, we found that the variance in the predicted versus true plots was comparable to the variance in the distance models shown in Figure 7.4. The CNNs are capable of handling both the non-linearity and the rapid change in the Lennard-Jones function as $r \to 0$. For the model trained on uniformly sampled energies, we found a 20% decrease in the root mean squared error (RMSE) when comparing the uniformly sampled model with the non-uniformly sampled model (i.e the model shown in Figure 7.4). We therefore conclude that the distance models perform better simply due to our sampling technique in the data generation process. For all of the dimer models in this manuscript, we uniformly sampled the distance, not the energy. When examining the distribution of input distances and energies, one can clearly see the non-uniformity in the distribution of energies for both the 2D and 3D models.

To compare with classic machine learning methods, we also performed tests using a multi-layer perceptron (MLP), kernel ridge regression (KRR), and random forests (RF) for 50,000 distances in the range $1 \leq r_{12} \leq 2$. To perform these tests, we used the scikit-learn [102] framework in Python. For KRR, we tried a linear and Gaussian kernel. For the linear kernel we used an alpha value of 1, and a degree 3 polynomial with a coefficient of 1. For the Gaussian kernel, we used $\gamma = 1, 2, 4$ and 16. We found that all of these parameters gave similar results for the MAE on the test set, which is reported in Table 7.1. For the MLP, we used 2 hidden layers consisting of 10 neurons, ReLU activation functions, the Adam optimization scheme, a learning rate of 0.001, and 200 epochs. For the RF model, we used 200 estimators (trees), the mean squared error to measure the quality of a split, and the maximum number of features was 256×256. The input to these models was the raw flattened images. RF performed best, with a MAE of 0.0168 Å, while MLP and KRR performed similarly, with a MAE of 0.124 Å. We found that the DNN outperforms all of these models with a MAE of 9.99×10^{-4} Å. The CNNAS approach for dimer molecules with raw data in both two and three dimensions is extremely accurate. It should be noted that the traditional machine learning models can be improved with feature engineering and

parameter optimization. As an example, Brockherde *et al.* [17] used KRR and were able to make energy predictions for a H_2 molecule within chemical accuracy with only 200 training examples. A DNN can only perform well with many training examples due to the large number of tuneable parameters. In the low volume data domain (e.g. only a few hundred training examples) a traditional machine learning model would be a more suitable choice. In the high volume data domain, DNNs are the suitable choice. The features are learned from the raw data, which avoids the feature engineering stage of constructing a machine learning model using a traditional approach.

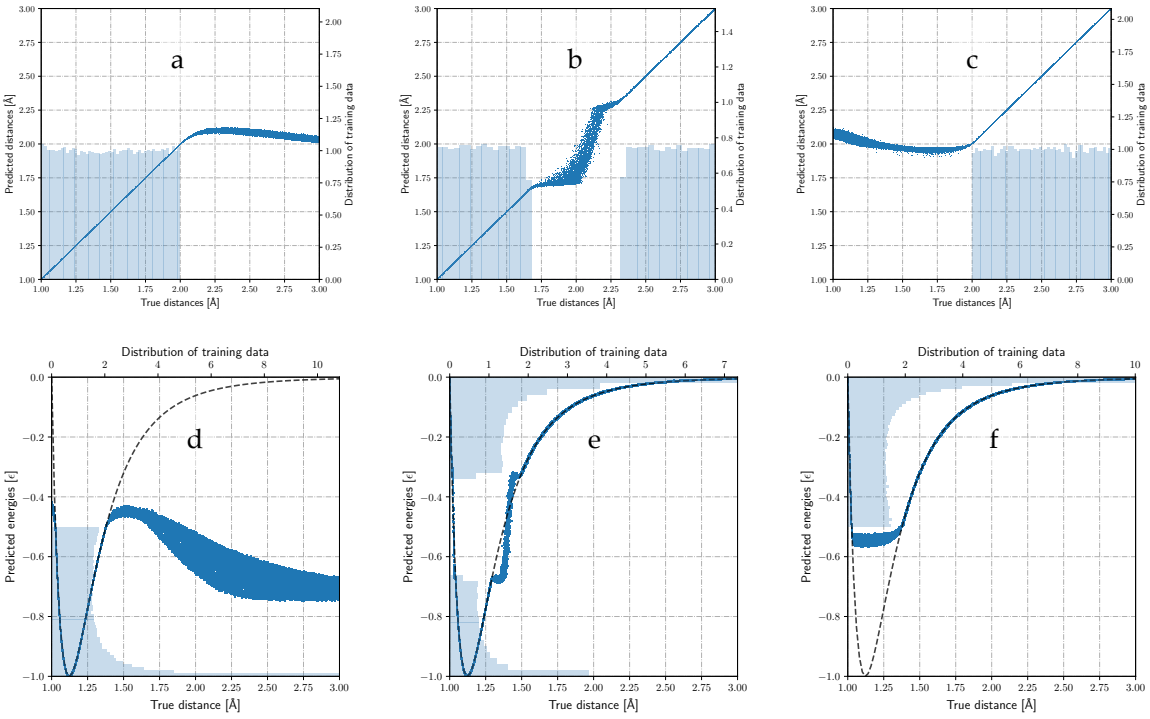


Figure 7.5: Predicted distance versus true distance (labelled a, b, and c) and predicted energy versus true distance (labelled d, e, and f) plots for the interpolation and extrapolation experiments. (a), (b), and (c) are the true distance versus predicted distance plots for the 2D distance models. (d), (e), and (f) are predicted energy versus true distance for to the 2D energy models. (a), (c), (d), and (f) correspond with extrapolation, and (b) and (e) correspond to interpolation, all described in subsection 7.3.2. The light blue distribution shown in the background of all plots gives the distribution of training data (a, b, and c correspond to distance distributions and d, e, and f correspond to energy distributions). The dotted grey lines in (d), (e), and (f) is the model Lennard-Jones curve, described by Equation 7.2.

7.3.2 Interpolation and Extrapolation

To investigate how well the CNNAS handles interpolating and extrapolating distances and energies, we constructed 2D and 3D models where we excluded training examples within a certain range. For distances we define the total range to be $1 \leq r_{12} \leq 3$, and for energies we define the total range to be $-1 \leq U(r_{12}) \leq 0$. When we perform an extrapolation test, we task the model to predict values that are greater or less than the range of the original dataset the model was trained on. When we perform an interpolation test, we task the model to predict values inside of the range it has been trained on, but in regions where it has not seen training examples. Although outside of the range it has been trained on, this range is within the minimum and maximum values of the original training dataset. We then trained 2D and 3D models for three different distance and energy ranges within their respective total ranges. Looking to Figure 7.5, we clearly see that all of the models fail when extrapolating. The models are only capable of predicting values within the range they have been trained on. Therefore, when the models are predicting values outside of this range, they return values on the endpoints of the ranges. To avoid this extrapolation issue, one must be aware that the model should see as large of a range as possible to make predictions for a general system. We found that in the interpolation tests, the model's ability to interpolate was also poor. Although these models were not trained on certain regions within the total ranges, they were able to make predictions within these regions. When tasked with interpolation, the MAE of the test set for the 2D distance interpolation model was 3.43×10^{-2} Å, which is a 1070% increase in comparison to MAE of the original corresponding test set. The original corresponding test set does not task the network to predict values outside of the range it has originally been trained on. The original test set has the same range as the training set. All of the error comes from the interpolation region. Similar effects were observed for the 2D and 3D distance models.

7.3.3 Hexagonal sheets

After concluding that our methodology allows for a successful DNN model to predict dimer distances and energies, we then moved on to much more complex many-body systems. As

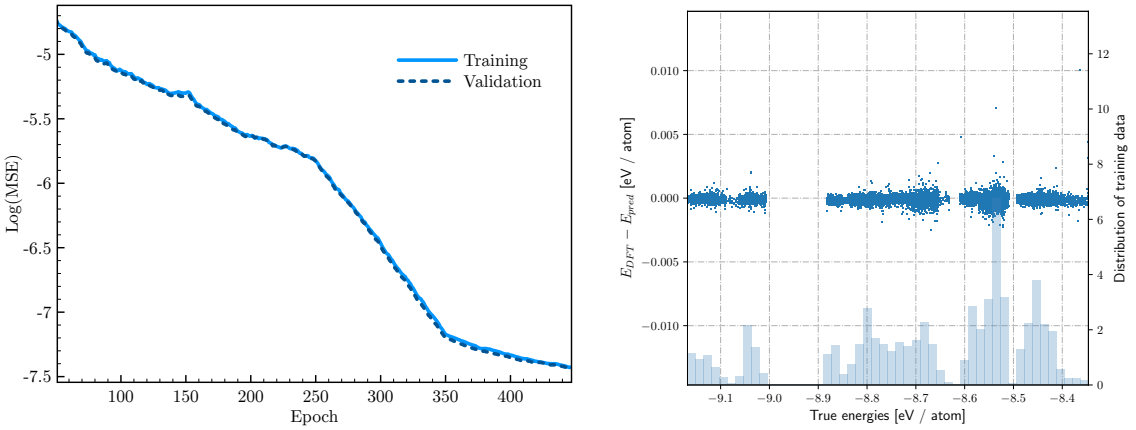


Figure 7.6: Left: Running average (with a window size of 50) of the training and validation loss during the training process. Right: DFT energy per atom minus the predicted energy per atom for the DFT model. The light blue distribution in the background gives the distribution of data used to train the model. The gap in the distribution arises from the absence of DFT energies within that particular range. At 300 epochs, the learning rate was lowered from 10^{-5} to 10^{-6}.

mentioned previously, we created an input data set for a DNN model by generating first principles molecular dynamics for a graphene sheet, a hBN sheet, and a graphene-hBN heterostructure. Within these structures, we also created either single point defects by removing one atom, or Stone-Wales defects by deforming the crystal lattice. After collecting the molecular dynamics data, we converted the molecular dynamics trajectories to images using Equation 7.1, and combined all of the images together into one data set. We first trained using network architecture 1 from Figure 7.2, but we found that the loss as a function of epoch did not decrease exponentially. Due to the increase of information (or number of atoms) within the images, the number of reducing convolutional layers eliminated information in the network necessary for making accurate energy predictions. This led to our choice of network architecture 2, also seen in Figure 7.2. This network architecture has fewer reducing convolutional layers, which allows for more information to be transmitted to the final fully connected layer. We found that this aided in the prediction process of DFT energies. We found the loss to decrease exponentially, indicating the successful training of this model. In Figure 7.6, we can clearly see that the network does exceptionally well at predicting DFT energies for trajectories it had not seen before. The MAE of the test

set was an impressive 0.0119 eV for total energies, or 0.198 meV / atom. Not only is the accuracy exceptional, but the computational cost was minimal. We were able to calculate approximately one hundred thousand total energies on the order of minutes.

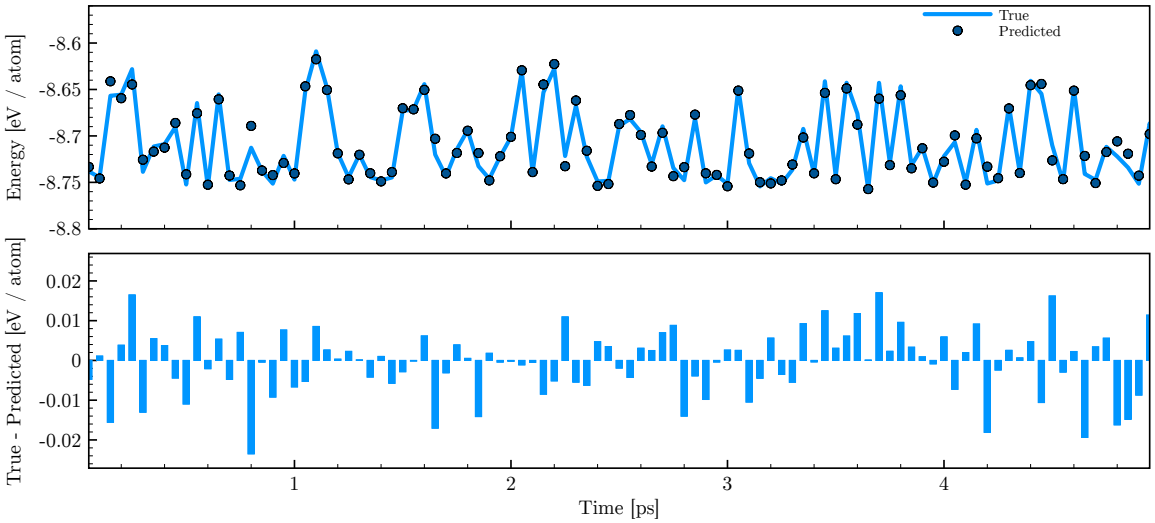


Figure 7.7: Top: True DFT and predicted energies per atom as a function of time for the hBN MD for every 100 steps. Bottom: True DFT energy per atom minus the predicted energy per atom as a function of time for the hBN MD for every 100 steps.

The viability and extensibility of a machine learned model can be shown by evaluating the model on its corresponding testing set, but a more rigorous test for a model is using it in practice. To further demonstrate the extensibility of our model, we performed additional MD calculations of the hexagonal-boron nitride surface and calculated predicted energies using the model for every 100 steps of the MD. For this newly generated MD we used the same parameters as before when generating the data, but we used a time step of 20.7 atomic units. We collected 5 ps of data for the new predictions. Looking to Figure 7.7, we plot the DFT and predicted energies as a function of time for every 100 MD steps. The MAE of the new predictions is 6.85 meV / atom. The maximum absolute error predicted for the MD trajectory is 38.8 meV / atom. This additional test further confirms the viability and extensibility of our machine learned model.

7.4 Conclusion

We have shown that our new methodology can be used successfully for predicting atomic distances and energies for dimer molecules as well as DFT energies for 2D hexagonal sheets. For dimer molecules, we found that our method is most accurate in 2D, which can be attributed to the increase of training data, and increased grid spacing in comparison to the 3D models. When testing the limits of our method, we found that the models are limited by the range of data they have been trained on. When extrapolating, the models predict values on the boundaries of the ranges they have been trained on. When testing the models' ability to interpolate, the models also make poor predictions. Although the models have not seen any data within the interpolation regions, they are still able to make predictions within the space spanned by the minimum and maximum values of the training set. Lastly, and most importantly, we found that CNNAS perform exceptionally well when tasked to predict DFT energies for a variety of hexagonal surfaces. The MAE we found was 0.0119 eV for the total energies, or 0.198 meV / atom for the test dataset. In addition, when the model was tasked to calculate energies of a new MD trajectory for hBN, it also succeeded with a MAE of 6.85 meV / atom.

7.5 Acknowledgement

The authors acknowledge NSERC, OGS, and SOSCIP for funding and computational resources, and NVIDIA for a faculty hardware grant.

Chapter 8

Deep Learning and Density Functional Theory

In this Chapter, we build on Chapter 7 by using a featureless approach with a deep learning architecture that is inherently extensive. In Chapter 7, a deep learning model was able to make predictions with errors smaller than chemical accuracy when compared to density functional theory (DFT) calculations. However, the model was restricted to perform inference on structures that had the same physical size during training. When constructing a machine learning model, the weight structure depends on the input dimensionality, and is "locked" to the size used during training. Consider a grey-scale image with dimensions 28×28. The weights of the first layer defined in the network depend on this input size, and therefore only images with dimensions 28×28 are allowed. In order to accelerate the design of nanoscale materials, one must have the capacity to simulate large system sizes rapidly. Here, we use extensive deep neural networks (EDNNs) [103] on 2D model systems, demonstrating the capability of featureless, image-based DNNs to predict quantities of interest in DFT.

Similar to Chapter 7, the models developed in this chapter can also be used in a design pipeline. If one is able to easily map the structure to the external potential, the energies and electron density could be computed rapidly. This would allow one to rapidly traverse through a design space. In addition, one could also have an energy or charge density in mind, and solve for input external potentials (and in turn structures that form those potentials) using automatic differentiation.

This work was published in Physical Review A in 2019.

Kevin Ryczko,
Department of Physics,
University of Ottawa

David Strubbe,
Department of Physics,
University of California, Merced

Isaac Tamblyn,
National Research Council of Canada,
Department of Physics, University of Ottawa

Abstract

We show that deep neural networks can be integrated into, or fully replace, the Kohn-Sham density functional theory scheme for multi-electron systems in simple harmonic oscillator and random external potentials with no feature engineering. We first show that self-consistent charge densities calculated with different exchange-correlation functionals can be used as input to an extensive deep neural network to make predictions for correlation, exchange, external, kinetic and total energies simultaneously. Additionally, we show that one can also make all of the same predictions with the external potential rather than the self-consistent charge density, which allows one to circumvent the Kohn-Sham scheme altogether. We then show that a self-consistent charge density found from a non-local exchange-correlation functional can be used to make energy predictions for a semi-local exchange-correlation functional. Lastly, we use a deep convolutional inverse graphics network to predict the charge density given an external potential for different exchange-correlation functionals and asses the viability of the predicted charge densities. This work shows that extensive deep neural networks are generalizable and transferable given the variability of

the potentials (maximum total energy range ≈ 100 Ha), because they require no feature engineering, and because they can scale to an arbitrary system size with an $\mathcal{O}(N)$ computational cost.

keywords: deep learning, density functional theory, convolutional neural networks, deep convolutional inverse graphics networks

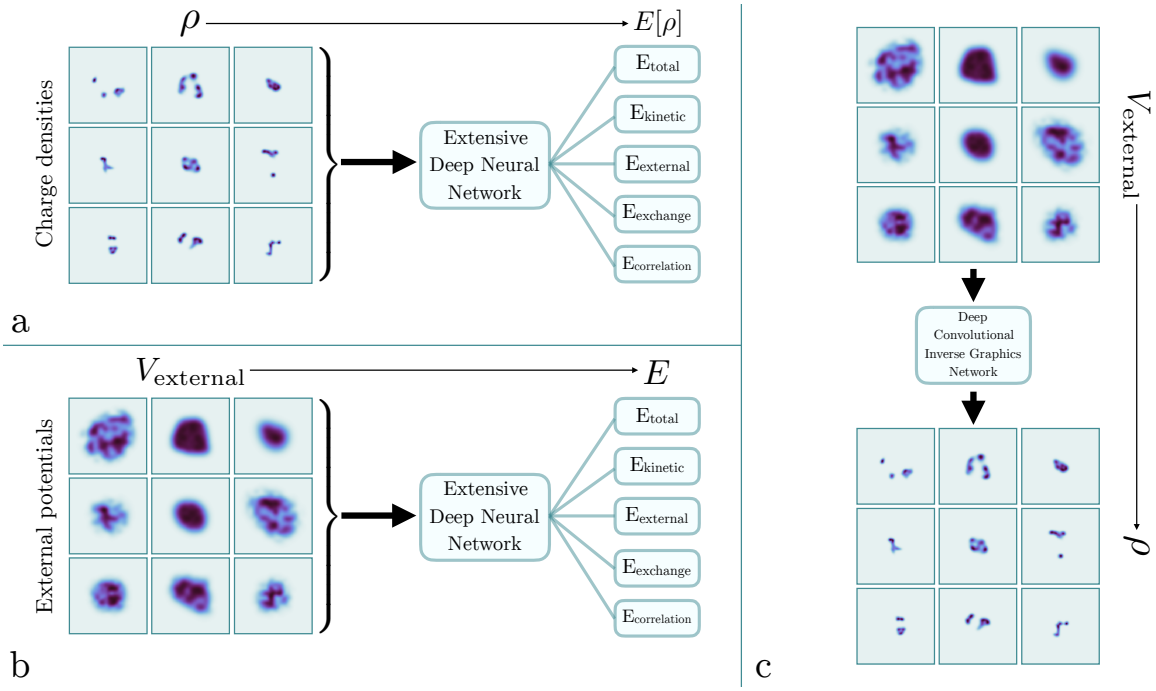


Figure 8.1: A graphical representation that outlines the objectives of this report. In a. and b. we show that both charge densities and external potentials can be used as input to extensive deep neural networks (EDNNs [103]) to predict the total, kinetic, external, exchange and correlation energies. The images shown are some of the random (RND) potentials along with the self-consistent charge density for that potential. In c. we show that deep convolutional inverse graphics networks (DCIGNs [104]) can be used to map external potentials to their respective self-consistent charge densities.

8.1 Introduction

Kohn-Sham (KS) density functional theory (DFT)[41] is the standard theoretical tool to study nanoscale systems. Despite its success, DFT calculations for atomistic systems containing tens of thousands to millions of atoms are exceptionally demanding from a computational perspective and are rare in the literature. Machine learning techniques can replace conventional DFT calculations to overcome this computational barrier. Machine learning models are ideal because they rival the accuracy of the method they are trained on, but can be less demanding to evaluate from a computational standpoint. There have been many reports where artificial neural networks (ANNs) have been used to represent potential energy surfaces to accelerate electronic structure calculations [105, 106, 107, 108, 11, 109, 25, 16]. These reports focus on feature engineering or defining some abstract representation of atomistic

systems allowing one to use an ANN. Instead, we focus our review on reports that avoid feature engineering and utilize the electron density in conjunction with machine learning. More specifically, machine learning has become a popular choice to represent energy functionals in DFT [14, 110, 111, 112, 113], or to completely circumvent the KS scheme [17, 20]. In deep learning, the machine learning model learns the hierarchical features during training rather than inputing abstract representations. Due to the large number of tuneable parameters in deep neural networks (DNNs) that may include a variety of layers (i.e. convolutional, fully connected, max pooling, etc.), there must be thousands (if not hundreds of thousands) of training examples to find a stable minima with an acceptable accuracy. Generating these training examples is a computationally expensive task, but a trained DNN can evaluate a given quantity at a fraction of a cost compared to the original method.

An alternative, novel approach that has been taken recently by Brockherde *et al.* [17] is to focus more on uniformly sampling the space that a machine learning model will eventually predict and to use traditional machine learning with far fewer tuneable parameters. This approach was successful in predicting KS-DFT total energies and charge densities in one dimension (1D) for random Gaussian potentials and for small molecules in three dimensions (3D). Due to the use of Kernel Ridge Regression (KRR), they were able to achieve an acceptable accuracy with a relatively small number of training examples. Unfortunately, KRR is known to have poor scaling with respect to the number of training examples, making it difficult to train with a large (and more diverse) set of training examples.

In KS-DFT, one of the contributions to the total energy is the non-interacting kinetic energy. Before the KS scheme was realized, Hohenberg and Kohn [40] postulated the formalism for an interacting kinetic energy functional of the density. An analytic expression for the exact interacting kinetic energy functional with respect to the electron density is unknown. This is one of the major downfalls of orbital-free (OF) DFT, where all energy contributions are explicitly written in terms of the electron density. This shortcoming provides motivation to construct an approximate functional of the density with a machine learning model. In a report done by Yao *et al.* [114], a convolutional neural network (CNN) was used to represent the kinetic energy functional in the OF-DFT total energy expression for various hydrocarbons. Their data generation process consisted of performing KS-DFT and collecting

the charge density along with the KS non-interacting kinetic energy. The charge density was then used as input to the CNN with the KS non-interacting kinetic energy as the label. With this representation they were able to successfully reproduce potential energy surfaces when compared to the true KS potential energy surfaces. In another report by Snyder *et al.* [112], they were able to use a machine learning model to make kinetic energy predictions given a charge density for a diatomic molecule. Using their framework, they were able to accurately dissociate the diatomic molecule, and compute forces suggesting that *ab initio* molecular dynamics could eventually be done via machine learning methods.

When representing the kinetic energy with a machine learning model in the OF scheme, one then becomes concerned with calculating the functional derivative of the machine learning model with respect to the density. In a report from Li *et al.* [111], they showed there is a trade-off between accuracy and numerical noise when taking the functional derivative of a machine learning model. Brockherde *et al.* [17] avoided this issue by training a machine learning model to learn the mapping between the potential and the electron density, avoiding the functional derivative.

In another recent report by Kolb *et al.* [113], a software package was developed to combine artificial neural networks with electronic structure calculations and molecular dynamics engines. Using their newly developed software, they were able to show that artificial neural networks can be used to make predictions with the electronic charge density as input and various energies as output. Specifically, they were able to predict energies and band gaps calculated at a higher level of theory from charge densities calculated at a lower level of theory. This approach is very advantageous as high level theory calculations (i.e. G_0W_0 [115]) become quite computationally expensive for larger systems.

Although significant progress has been made incorporating machine learning and deep learning to a variety of electronic structure problems, most do not have the ability to properly handle extensive properties. In some of our past work, we introduced extensive DNNs (EDNNs) [103] to intrinsically learn extensive properties. This means that when the DNN learns the fundamental screening length scale it can then easily scale up to massive systems in a trivially parallel manner. EDNNs work by first dividing up an image into fragments

which are called focus regions. These fragments are then padded with context regions. The context regions may also respect periodic boundary conditions. Each of these fragments can then be simultaneously passed into machine learning models that share weights. It should be noted here that any machine learning method that uses back propagation to minimize the loss function can be used. Finally, the outputs of the machine learning models are then summed yielding the final prediction from the EDNN.

In this report, we show that EDNNs have the capability to learn energy and charge density mappings that could replace some, if not all, calculations in KS-DFT scheme. We push the frontier of what EDNNs can learn from charge densities and external potentials by calculating the self-consistent charge densities in external potentials with extreme variabilities. In previous reports that focus on small molecules [17, 108, 113, 114], the self-consistent charge densities generated from molecular dynamics are similar and have small energy ranges (i.e. ≈ 31.8 mHa [17]). We avoid small molecules (where the charge density would be localized in space), and truly challenge the ability of EDNNs to make accurate predictions across a variety of electronic environments. Quantitatively speaking the energy range of our 10 electron calculations with our random (RND) external potentials is ≈ 100 Ha. This report is outlined as follows: In Section 8.2, we describe our data generation process, as well as the DNN topologies and hyper-parameter selections. In Subsection 8.3.1, we show that DNNs have the capability to act as density functionals and can accurately predict the exchange, correlation, external, kinetic, and total energies *simultaneously* (Subsection 8.3.1). We also show that EDNNs can also circumvent the KS scheme (Subsection 8.3.1) by mapping the external potential to all of the aforementioned energies simultaneously. Additionally, we show that EDNNs can be used in a somewhat "perturbative" manner, where we predict energies computed with semi-local or non-local exchange-correlation functionals from non-local electron densities. In Subsection 8.3.2, we show that deep convolutional inverse graphics networks (DCIGNs) can also map the external potential to the electron density, and assess the viability of the predicted electron density. Lastly, in Section 8.4, we summarize our results and consider future work that could be done with our new framework. The outline of this manuscript can be seen graphically in Figure 8.1.

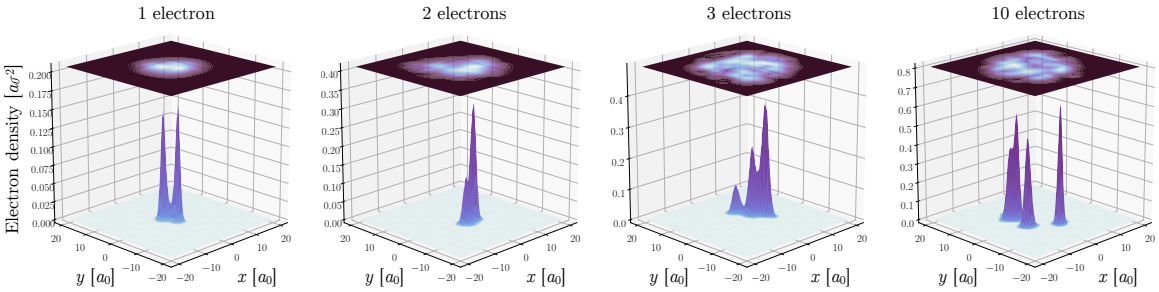


Figure 8.2: Computed charge densities (3D surfaces) and random external potential energy surfaces (2D surfaces) for typical configurations of systems with 1, 2, 3, and 10 electrons.

8.2 Methods

We investigate two-dimensional (2D) electron gases within the KS-DFT framework [41] for two external potentials: simple harmonic oscillator (SHO) and RND. These external potentials have been used in a previous study [116] for direct diagonalization, one-electron calculations. In the KS-DFT framework, one minimizes the total energy functional

$$E[\rho] = T[\rho] + E_{\text{ext}}[\rho] + E_{\text{Hartree}}[\rho] + E_{\text{XC}}[\rho] \tag{8.1}$$

which leads to the expression

$$\begin{aligned} E[\rho(\mathbf{r})] &= 2\sum_{i}^{N/2} \epsilon_i - \frac{1}{2}\int\int d\mathbf{r}d\mathbf{r}' \, \frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} \\ &+ E_{\text{XC}}[\rho] - \int d\mathbf{r} \; \mu_{\text{xc}}(\rho(\mathbf{r}))\rho(\mathbf{r}). \end{aligned} \tag{8.2}$$

In Equation 8.1, T is the non-interacting kinetic energy, E_{ext} is the energy due to the interaction of the electrons with the external potential, E_{Hartree} is the electrostatic energy describing the electron-electron interactions, E_{XC} is the exchange-correlation energy, and $\mu_{xc}(\rho(\mathbf{r}))$ is the same as defined in [41].

Using EDNNs, we investigate the feasibility of learning the total energy as well as the individual contributions to the total energy. We therefore have trained models to predict the total, non-interacting kinetic, external, exchange, and correlation energies. The external

potentials chosen for this report, as mentioned previously, are SHO and RND potentials in 2D. The SHO potentials take the form

$$V_{\text{ext}}(\{x_i\}) = \frac{1}{2}\sum_{i}^{D} k_i(x_i - x_{0_i})^2 \tag{8.3}$$

where D is the dimension, $k_i = m\omega_i^2$ is the spring constant, and x_{0_i} is the shift of the potential in a given coordinate. For the RND potentials we follow the work of Mills *et al.* [116] when generating the potentials on a grid. We refer the reader to the original manuscript [116] for more information on the RND potential generation. To briefly summarize the process, the first step consists of generating a random binary image of 0's and 1's and then applying a gaussian filter. The second step consists of generating a mask that is constructed with a random convex hull and an additional gaussian blur. The mask is then applied onto the image yielding the final result. The larger energy scale of the RND external potentials can be attributed to the length scales of the RND external potentials. The average Gaussian kernel sizes in the external potential generation is ~ 3 Bohr, whereas the average length scale of the SHO external potentials is ~ 12 Bohr. Assuming that the energy scales as $E \sim 1/r^2$, the energy scale of the RND external potentials is 16 times larger than the SHO external potential energy scale on average. To create datasets large enough to use DNNs, we chose to randomly sample k_i and x_{0_i} such that $0.01 \leq k_i \leq 0.16$ Ha$/a_0^2$ (Hartree per Bohr^2) and $-8.0 \leq x_{0_i} \leq 8.0\ a_0$. With a given selection of these variables, the external potential was then evaluated on a $40 \times 40\ a_0$ space with a 256×256 grid point mesh. We then chose to place either N =1, 2, 3, or 10 electrons in the 2D space. For each choice of the number of electrons, we generate an external potential, and then perform three DFT calculations, each with different exchange-correlation functionals. We used the local density approximation (LDA) exchange-correlation functional [117, 118], the Perdew-Burke-Ernzerhof (PBE) [44] functional for exchange and the LDA correlation functional, and the meta-generalized gradient approximation (MGGA) exchange functional from Pittalis *et al.* [119] and the LDA correlation functional. Here, we do not take the orbitals or charge densities from the LDA calculations to calculate energies at the PBE or MGGA level. All of the energies for each functional are calculated independently. All of the calculations were carried in

real space with the Octopus code [120, 121, 122]. For testing, we set aside 10% of each data set. This made for 90,000 training configurations and 10,000 testing configurations for each case of potential, number of electrons, and exchange-correlation functional. Note that the number of electrons and the type of external potential uniquely defines a dataset. Therefore, the 10% set aside for testing includes 10,000 external potentials and 30,000 charge densities (10,000 for each exchange-correlation functional choice). In addition, the labels were normalized independently such that each of the components had a range from zero to one. The calculations are summarized in Table 8.1 of Appendix 8.A.1. All data used in this report is available online (`http://clean.energyscience.ca/datasets`) along with the code (`https://github.com/kryczko/ednn`) to allow for future development of featureless deep learning based functionals.

When constructing the EDNNs, we used a mixture of Tensorflow [100] and TFLearn [123] in Python. For the networks topologies we build on our previous reports [20, 103] and use EDNNs where each tile of the EDNN has the same in-tile CNN used previously for predicting KS-DFT total energies of 2D hexagonal sheets [20]. For clarity, the in-tile CNN consisted of 2 reducing convolutional layers with kernel sizes of 3, 6 non-reducing convolutional layers with kernel sizes of 4, 1 reducing convolutional layer with a kernel size of 3, 4 non-reducing convolutional layers with kernel sizes of 3, a fully connected layer with 1024 neurons, and a final fully connected layer with one neuron. All of the activations used were rectified linear units. We emphasize that in our approach, we do not do any sort of feature engineering, like past reports that use ANNs [109, 11, 108, 107, 106, 105, 113]. The convolutional layers in the EDNNs identify relevant features during the training process. When utilizing an EDNN, one must declare the focus and context regions which is used to "tile" up the image into fragments. To find the ideal focus and context regions, we started by training the EDNNs on the 2D charge density to total energy mapping as well as the 2D external potential to total energy mapping for the 1, 2, 3, and 10 electron systems for calculations done with the LDA exchange-correlation functional and the SHO external potential. We chose a variety of focus and context sizes, and found that the optimal focus and context sizes are 128 pixels for the focus size, and 32 pixels for the context size. Our decision was based on a balance between accuracy and computation time. A larger focus

size lowers the computation time, and a larger context size yields larger images, resulting in more neurons in the EDNNs thereby improving the accuracy of the model. For a focus of 128 pixels, we found that the accuracies were very similar for various context sizes and the choice of 32 pixels was almost arbitrary. This hyperparameter search was the most computationally demanding task for this work due to the number of models that had to be trained. While training, we used a learning rate of 10^{-4} for 500 epochs when using the charge densities as input and a learning rate of 10^{-5} for 500 epochs when using the external potentials as input. In both cases, we further reduced the learning rates by a factor of 10 and trained for an additional 100 epochs. For clarity, an epoch is defined to be when the weights of the network have been updated for the entire training dataset once.

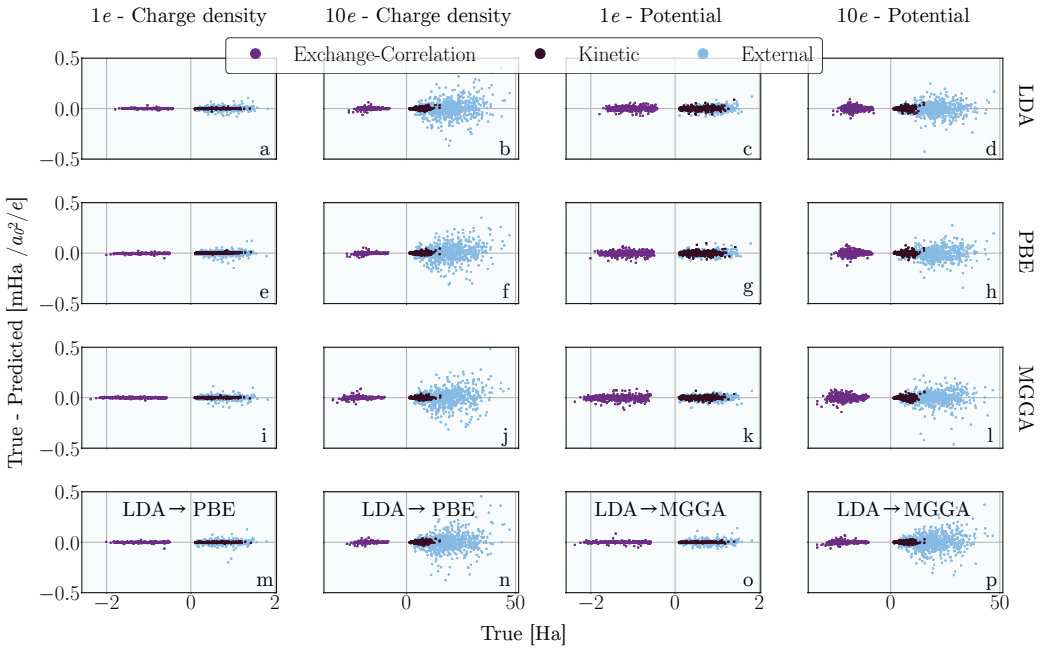


Figure 8.3: True minus predicted (in mHa / a_0^2 / electron) versus true (Ha) for various models with the RND potentials. Plots a-d are models trained with the LDA exchange-correlation functional, e-f with the PBE exchange-correlation functional, and i-l with the MGGA exchange-correlation functional. First column (a, e, i) is for 1 electron models where the charge densities were used as input. Second column (b, f, j) is for 10 electron models where the charge densities were used as input. Third column (c, g, k) is for 1 electron models where the external potentials were used as input. Fourth column (d, h, l) is for 10 electrons models where the external potentials were used as input. The bottom row (m-p) is for models where LDA charge densities were used as input, and the labels were either PBE energies (m, n) or MGGA energies (o, p). Plots m, o are for the 1 electron systems, and n, p for the 10 electron systems. It should be noted that one model is predicting the correlation, exchange, external, kinetic, and total energies. We have combined the exchange-correlation error and omitted the total energy error for clarity.

8.3 Results

8.3.1 Energy predictions

EDNNs as a functional

Firstly, we show that EDNNs can be used as an energy functional for correlation, exchange, external, kinetic and total energies. For the LDA, PBE, and MGGA functionals discussed in Section 8.2, we used the computed self-consistent charge densities as input to an EDNN and were able to successfully predict the correlation, exchange, external and total energies simultaneously for both SHO and RND external potentials. Starting with the models where the SHO external potentials were used in the DFT calculations, we found that the mean absolute errors for each particular case are less than 1.5 mHa. These can be seen in Table 8.2 of Appendix 8.A.2. In Figure 8.3, we show predicted minus true versus true for the one and ten electron models with the different exchange-correlation functionals when the RND external potentials were used in the DFT calculations. In this Figure, it is clear that the error of the models increase with the number of electrons. This increase in error is expected due to the increase in the range of energies and can be physically attributed to the increase of interactions in the system. Looking to Table 8.3 of Appendix 8.A.3 we also observe that the mean absolute errors become larger as the complexity of the exchange-correlation functional increases. In addition to these trends, we also notice that the energy with the largest mean absolute error comes from the external energy functional. This again can be attributed to the ranges of the various energies. The external energy has the largest range of all the energies being predicted. We also address the generalizability of the models by testing the model that was trained on 10 electron charge densities calculated with the RND external potentials and the LDA functional with 10 electron charge densities calculated with the SHO external potentials and the LDA functional (and vice versa). We found in both cases the errors increased by several orders of magnitude. This is not surprising given the different energy ranges of the datasets. On the contrary, when examining the true versus predicted plot for the model trained on the RND dataset but tested on with the SHO dataset, we found that a constant shift could simply be added to substantially decrease the error. We expect

that this constant shift could be easily rectified if SHO training examples are included in the training process.

DFT is a more popular choice for larger systems relative to wavefunction based methods because the exchange-correlation functionals used are computationally inexpensive relative to methods that employ exact exchange, for example. In light of this, we have trained EDNNs to predict energies at the PBE and MGGA level given a self-consistent charge density computed with the LDA exchange-correlation functional. In Figure 8.3, we consider 1 and 10 electron models trained on the mapping between LDA charge densities and either PBE or MGGA energies. Similar to the results mentioned above, the mean absolute errors increase both with the number of electrons and the complexity of the exchange-correlation functional. In Table 8.3 of Appendix 8.A.3, we also notice that the highest mean absolute error is for the external and total energies. This result further suggests that there is not a fundamental problem with learning the external energy, but the larger range of energies makes it more difficult for a EDNN to handle with extreme precision. In addition, since the correlation functional is the same across all of the calculations and the same testing data was used for each case of number of electrons, we can determine if the networks are learning the correlation energy in a similar manner. In Table 8.3, we can see that the correlation energies have similar magnitudes of error indicating that similar correlation functional mappings are being learned as one should expect. The success of learning the energies of a more accurate exchange-correlation functional given a less accurate charge density shows promise for other applications. A future application could include learning a G_0W_0 total energy from a DFT computed self-consistent charge density, similar to the work that was completed by Kolb *et al.* [113].

A note should be made about Table 8.3 with respect to the magnitude of some of the mean absolute errors reported. In comparison to the report by Mills *et al.* [116], some of the mean absolute errors are larger by some cases a factor of 10. In addition, the focus and context hyperparameters were optimized for the SHO external potentials. In the work of Brockherde *et al.* [17], they managed to reach chemical accuracy using three dimensional charge densities, but the energy range of their training set was ≈40 kcal/mol (for a benzene molecule). For their best reported model with a mean absolute error of 0.28 kcal/mol, the

relative mean absolute error, that we define as the mean absolute error divided by the range of the dataset is 0.007. In our 10 electron model with the RND external potential, our energy range was ≈100 Ha (62750.9 kcal/mol) and the mean absolute error of the total energy predictions was 78.514 mHa yielding a relative mean absolute error of 7.85×10^{-4}.

Circumventing Kohn-Sham DFT

In addition to using EDNNs as a functional, it is arguably more convenient to train a EDNN to learn the mapping between the external potential and the contributing energies of that system. It is more convenient because it avoids calculating a self-consistent charge density with the KS scheme. We have trained EDNNs to predict the exchange, correlation, external, kinetic, and total energy simultaneously using the external potential as input rather than the charge density. Again, in Figure 8.3 we show true minus predicted versus true for the correlation, exchange, external, kinetic, total energies for the RND external potentials. Here, it is evident that the charge density is more optimal as an input to an EDNN for the 1 electron systems. There is much more spread in the distribution when using external potentials as input compared to charge densities. For 10 electrons, this is not the case. Looking to Table 8.3 of Appendix 8.A.3, we can see that for 1, 2, and 3 electrons no matter what choice of exchange-correlation functional, it is less difficult to learn the mapping between $\rho \rightarrow E$ than $V \rightarrow E$. The mean absolute errors are lower for all energies. In the case of 10 electrons, the mean absolute errors in the external and total energies are lower for the models that have potentials as input. Although the errors are lower for the external and total energies, the mean absolute errors for correlation, exchange, and kinetic energies are larger. When training a model on a set of energies, there is a balance between the errors of the energies since the loss function depends on the sum over the mean squared errors between the true and predicted energies. In the case of using charge densities as input to the EDNN, we found the exchange, correlation, and kinetic energies can be predicted with much better accuracy than the external or total energies. In the case of using potentials as input to the EDNN, we found that there is more of a balance of accuracy between the different energies being predicted.

8.3.2 Image predictions with Deep Convolutional Inverse Graphics Networks

In both KS-DFT and OF-DFT, the self-consistent charge density is the central quantity that one is interested in calculating. Once one has the charge density, most other quantities can be calculated in a straightforward manner. In this Subsection, we address the viability of using DCIGNs to map the external potential to the self-consistent charge density in 2D for the RND potentials with the LDA, PBE, and MGGA exchange correlation functionals. DCIGNs were recently introduced in the literature [104] and have a similar topology to autoencoders [59]. The DCIGN that we have used has 4 reducing convolutional layers, 3 non-reducing convolutional layers, and 4 deconvolutional layers such that the output image has the same dimensionality as the input image. This topology differs slightly from the original work on DCIGNs [104], where a fully connected layer would replace our 3 non-reducing convolutional layers. Additionally, our DCIGN is deterministic. In the original work [104], random noise is introduced in the decoder to create a non-deterministic generative model. All of our convolutional layers use a kernel size of 3 with rectified linear unit activations. We used a learning rate of 10^{-5} while training for 500 epochs and dropped the learning rate by a factor of 10 before training for an additional 100 epochs. For this discussion we focus solely on the 10 electron calculations with the RND external potentials. We argue that these are the most challenging calculations to train with a DCIGN, and can therefore safely assume that the less complex calculations would be successful given the success of the most complex cases. In Figure 8.4, we show some of the predictions that the DCIGN made for 10 electrons calculations with the LDA exchange-correlation functional. There is a remarkable resemblance between the true (ρ_{true}) and predicted ($\rho_{\text{predicted}}$) charge densities. The DCIGN is capable of handling the extreme variability of the complex shapes, and is capable of handling the cases where the charge density is not isolated to one region of space. From a qualitative perspective, the DCIGN makes accurate predictions of the charge densities given RND external potentials.

Normally, when addressing the viability of a machine learning model from a quantitative perspective one considers the mean absolute error on the test set. We argue that a more rigorous test for $\rho_{\text{predicted}}$ would be mean absolute error of the energies associated with

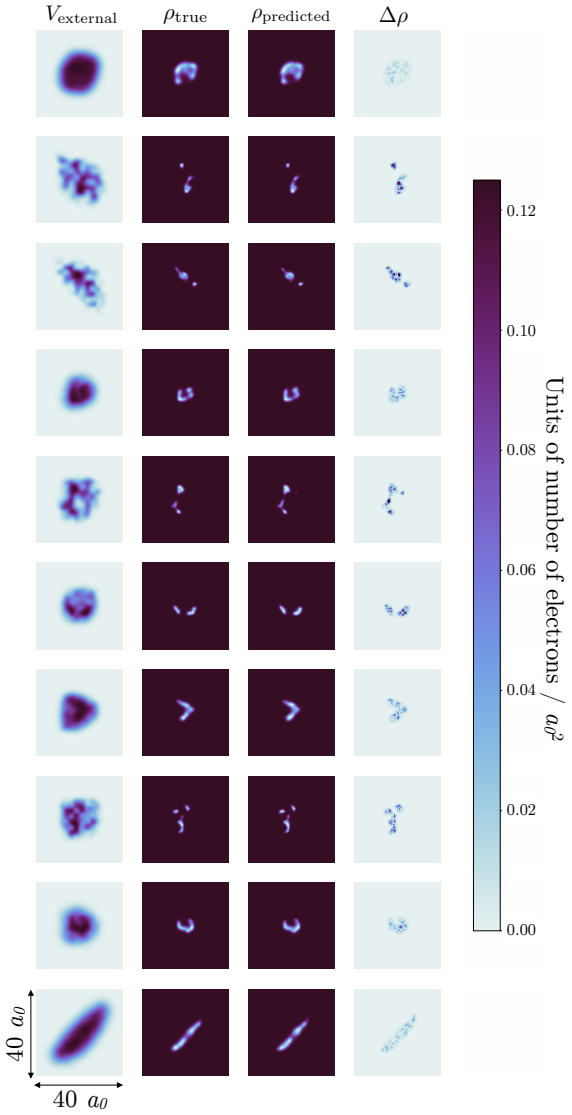


Figure 8.4: Examples of the RND potentials, true charge densities (ρ_{true}), predicted charge densities ($\rho_{\text{predicted}}$), and differences between the true and predicted charge densities ($\Delta\rho$). The charge densities shown here were computed with the LDA exchange-correlation functional. The colour bar is for the charge density differences $\Delta\rho$.

$\rho_{\text{predicted}}$. We therefore take $\rho_{\text{predicted}}$ and renormalize them such that $\int d\mathbf{r}\ \rho(\mathbf{r})_{\text{predicted}} = 10$. Afterwards, we use the renormalized $\rho_{\text{predicted}}$ as input to a subset of the models described in Subsection 8.3.1. We then compare the energies predicted from $\rho_{\text{predicted}}$ with the true energies. In Table 8.3 of Appendix 8.A.3, we show the mean absolute errors between the true and predicted energies for the different exchange-correlation functionals. When comparing the mean absolute errors of the predicted energies for $\rho_{\text{predicted}}$ with the energy predictions

made from the ρ_{true}, the minimal difference was seen for the correlation energies with a value of $\sim$ 6 mHa. This was true for all exchange-correlation functionals considered in this work. The maximal difference between the mean absolute errors when comparing the energy predictions of $\rho_{\text{predicted}}$ and ρ_{true} was the total energy which was $\sim$ 20 mHa. Again, this is true for all exchange-correlation functionals considered. In addition to this metric, we also report the mean absolute error for a model mapping $\rho_{\text{predicted}}$ to the true energies. We find an increase in the errors in comparison to the models that map the true charge densities to the true energies, but the errors are comparable to the errors when we evaluate $\rho_{\text{predicted}}$ with the model trained on the true charge densities. Training with the charge densities and energies as labels yields similar results. We also report the density driven error (DDE) using the same definition as Brockherde *et al.* [17] in Table 8.3. We find similar trends in the DDEs when comparing them to the mean absolute errors of $\rho_{\text{predicted}}$. In addition, to compare with Brockherde *et al.* [17], our relative mean absolute error is a factor of $\approx$ 7 times smaller.

8.4 Conclusion

In conclusion, we have shown that EDNNs and DCIGNs can be used alongside, or replace conventional KS-DFT calculations. For both the RND and SHO external potentials, EDNNs have the capability to make highly accurate energy predictions using both the charge densities and the external potentials as input for correlation, exchange, external, kinetic and total energy simultaneously (dataset is available online here: `http://clean.energyscience.ca/datasets`). In addition, we have shown that DCIGNs have the capability to predict charge densities given an external potential. Qualitatively speaking, the predicted charge densities are remarkably similar to the true charge densities. Quantitatively speaking, the relative mean absolute errors were found to be smaller than previous, state-of-the-art work [17]. The results of this report show promise for future application in two regards. First, that this framework has the capability to make predictions of higher level theory calculations given a lower level theory charge density similar to a previous report [113]. Second, both EDNNs and DCIGNs can be used to calculate energies covering a large range of electronic environments to a high level of accuracy.

8.5 Acknowledgements

The authors acknowledge funding from the Natural Sciences and Engineering Research Council of Canada, and Compute Canada and SOSCIP for computational resources. The authors would also like to thank NVIDIA for a faculty hardware grant.

8.A Supplemental Information

8.A.1 More details on the generated data

N	V_{ext}	$V_{\text{X}} + V_{\text{C}}$	Number of calculations
1	SHO	LDA + LDA	100,000
1	SHO	PBE + LDA	100,000
1	SHO	MGGA + LDA	100,000
1	RND	LDA + LDA	100,000
1	RND	PBE + LDA	100,000
1	RND	MGGA + LDA	100,000
2	SHO	LDA + LDA	100,000
2	SHO	PBE + LDA	100,000
2	SHO	MGGA + LDA	100,000
2	RND	LDA + LDA	100,000
2	RND	PBE + LDA	100,000
2	RND	MGGA + LDA	100,000
3	SHO	LDA + LDA	100,000
3	SHO	PBE + LDA	100,000
3	SHO	MGGA + LDA	100,000
3	RND	LDA + LDA	100,000
3	RND	PBE + LDA	100,000
3	RND	MGGA + LDA	100,000
10	SHO	LDA + LDA	100,000
10	SHO	PBE + LDA	100,000
10	SHO	MGGA + LDA	100,000
10	RND	LDA + LDA	100,000
10	RND	PBE + LDA	100,000
10	RND	MGGA + LDA	100,000
total			2,400,000

Table 8.1: Summary of the calculations that were used for training and testing the deep learning models. N is the number of electrons, V_{ext} is the external potential chosen (see text), and $V_{\text{X}} + V_{\text{C}}$ are the exchange-correlation potentials chosen. Note that the combination of the number of electrons and external potential produces a unique data set. For example, the 3 electron systems in RND potentials has 100,000 external potentials but contributes 300,000 calculations due to the use of different exchange-correlation functionals.

8.A.2 Mean absolute errors with the simple harmonic oscillator external potentials

$N_{\text{electrons}}$	Input	Functional	$E_{\text{correlation}}$	E_{exchange}	E_{external}	E_{kinetic}	E_{total}
1	ρ	LDA	0.1 (0.1)	0.1 (0.1)	0.1 (0.1)	0.1 (0.1)	0.2 (0.2)
2	ρ	LDA	0.1 (0.1)	0.1 (0.2)	0.1 (0.2)	0.1 (0.1)	0.3 (0.4)
3	ρ	LDA	0.1 (0.1)	0.1 (0.2)	0.2 (0.3)	0.1 (0.2)	0.5 (0.6)
10	ρ	LDA	0.1 (0.1)	0.2 (0.2)	0.3 (0.6)	0.2 (0.3)	0.9 (1.4)
1	ρ	PBE	0.1 (0.1)	0.2 (0.2)	0.1 (0.2)	0.1 (0.1)	0.2 (0.3)
2	ρ	PBE	0.1 (0.1)	0.2 (0.2)	0.1 (0.2)	0.1 (0.1)	0.3 (0.4)
3	ρ	PBE	0.1 (0.2)	0.2 (0.3)	0.3 (0.4)	0.2 (0.2)	0.8 (0.9)
10	ρ	PBE	0.1 (0.1)	0.2 (0.2)	0.4 (0.6)	0.2 (0.2)	0.9 (1.5)
1	ρ	MGGA	0.1 (0.1)	0.2 (0.3)	0.1 (0.2)	0.1 (0.1)	0.3 (0.3)
2	ρ	MGGA	0.2 (0.2)	0.3 (0.4)	0.2 (0.2)	0.1 (0.2)	0.4 (0.5)
3	ρ	MGGA	0.1 (0.1)	0.2 (0.3)	0.2 (0.2)	0.1 (0.2)	0.4 (0.5)
10	ρ	MGGA	0.2 (0.3)	0.4 (0.5)	0.5 (0.8)	0.3 (0.4)	1.3 (1.9)
1	V_{ext}	LDA	0.1 (0.1)	0.1 (0.1)	0.1 (0.1)	0.1 (0.1)	0.2 (0.2)
2	V_{ext}	LDA	0.1 (0.1)	0.1 (0.2)	0.1 (0.1)	0.1 (0.1)	0.2 (0.3)
3	V_{ext}	LDA	0.1 (0.1)	0.1 (0.2)	0.1 (0.2)	0.1 (0.1)	0.3 (0.4)
10	V_{ext}	LDA	0.1 (0.2)	0.2 (0.3)	0.3 (0.8)	0.2 (0.3)	0.8 (1.1)
1	V_{ext}	PBE	0.1 (0.1)	0.1 (0.1)	0.1 (0.1)	0.1 (0.1)	0.1 (0.2)
2	V_{ext}	PBE	0.1 (0.1)	0.1 (0.2)	0.1 (0.1)	0.0 (0.1)	0.2 (0.3)
3	V_{ext}	PBE	0.1 (0.1)	0.1 (0.2)	0.1 (0.2)	0.1 (0.1)	0.3 (0.4)
10	V_{ext}	PBE	0.1 (0.2)	0.2 (0.4)	0.4 (0.8)	0.2 (0.3)	0.8 (1.1)
1	V_{ext}	MGGA	0.1 (0.1)	0.1 (0.2)	0.1 (0.1)	0.1 (0.1)	0.2 (0.2)
2	V_{ext}	MGGA	0.1 (0.1)	0.2 (0.2)	0.1 (0.1)	0.0 (0.1)	0.2 (0.3)
3	V_{ext}	MGGA	0.1 (0.1)	0.2 (0.3)	0.1 (0.2)	0.1 (0.1)	0.3 (0.4)
10	V_{ext}	MGGA	0.1 (0.2)	0.3 (0.5)	0.3 (0.7)	0.2 (0.3)	0.7 (1.0)
1	ρ	LDA→PBE	0.1 (0.1)	0.1 (0.2)	0.1 (0.1)	0.1 (0.1)	0.2 (0.3)
2	ρ	LDA→PBE	0.1 (0.1)	0.2 (0.2)	0.1 (0.2)	0.1 (0.1)	0.3 (0.4)
3	ρ	LDA→PBE	0.1 (0.1)	0.1 (0.2)	0.2 (0.2)	0.1 (0.2)	0.4 (0.5)
10	ρ	LDA→PBE	0.1 (0.2)	0.2 (0.2)	0.3 (0.6)	0.2 (0.3)	0.8 (1.4)
1	ρ	LDA→MGGA	0.1 (0.1)	0.2 (0.3)	0.1 (0.2)	0.1 (0.2)	0.3 (0.3)
2	ρ	LDA→MGGA	0.1 (0.2)	0.3 (0.3)	0.1 (0.2)	0.1 (0.2)	0.3 (0.5)
3	ρ	LDA→MGGA	0.1 (0.2)	0.3 (0.4)	0.2 (0.3)	0.2 (0.2)	0.5 (0.7)
10	ρ	LDA→MGGA	0.1 (0.2)	0.2 (0.3)	0.4 (0.7)	0.2 (0.3)	0.9 (1.5)

Table 8.2: Mean absolute errors (in mHa per electron) and root mean squared errors (in parenthesis) for models trained in this report for the SHO potentials. The abbreviations ρ, and V_{ext} are charge density, and potential respectively. The arrows (i.e. LDA→PBE) indicate that the charge density used as input to the DNN was calculated using the LDA exchange-correlation functional, but the labels (energies) were calculated using another exchange-correlation functional.

8.A.3 Mean absolute errors with the RND external potentials

$N_{\text{electrons}}$	Input	Functional	$E_{\text{correlation}}$	E_{exchange}	E_{external}	E_{kinetic}	E_{total}
1	ρ	LDA	0.9 (1.5)	1.4 (2.3)	14.5 (31.7)	2.5 (3.7)	14.5 (31.1)
2	ρ	LDA	1.1 (2.0)	1.8 (3.1)	9.6 (19.4)	2.2 (3.4)	9.9 (25.4)
3	ρ	LDA	2.0 (3.5)	3.1 (5.4)	35.1 (57.0)	5.6 (8.6)	36.1 (58.4)
10	ρ	LDA	2.0 (3.8)	3.0 (5.9)	73.4 (117.1)	6.7 (10.2)	74.4 (119.4)
1	ρ	PBE	1.9 (2.5)	3.1 (4.0)	15.3 (32.2)	3.8 (4.9)	15.1 (31.2)
2	ρ	PBE	1.3 (2.1)	1.9 (3.1)	9.5 (19.3)	2.2 (3.1)	10.1 (21.7)
3	ρ	PBE	2.0 (3.3)	2.9 (4.9)	34.4 (55.6)	5.2 (7.7)	35.6 (56.8)
10	ρ	PBE	1.9 (3.6)	2.8 (5.5)	73.7 (115.0)	7.6 (11.4)	75.0 (117.1)
1	ρ	MGGA	1.0 (1.7)	2.1 (3.4)	13.7 (30.7)	2.5 (3.8)	13.9 (31.0)
2	ρ	MGGA	1.3 (2.2)	2.6 (4.6)	10.0 (17.8)	2.4 (3.9)	10.2 (17.8)
3	ρ	MGGA	1.8 (2.8)	3.6 (5.7)	34.1 (55.7)	4.7 (7.1)	36.5 (62.2)
10	ρ	MGGA	2.2 (4.1)	4.4 (8.4)	77.2 (122.1)	7.3 (11.3)	78.5 (126.1)
1	V_{ext}	LDA	5.3 (9.0)	8.6 (14.4)	17.8 (28.7)	11.5 (19.2)	22.3 (36.8)
2	V_{ext}	LDA	5.3 (8.7)	8.5 (13.6)	18.5 (31.8)	8.8 (13.5)	21.2 (33.4)
3	V_{ext}	LDA	10.6 (15.0)	16.8 (23.7)	46.2 (70.2)	15.8 (23.6)	43.2 (66.7)
10	V_{ext}	LDA	6.6 (9.9)	10.3 (15.4)	50.3 (86.1)	10.5 (16.7)	40.9 (73.2)
1	V_{ext}	PBE	6.0 (9.9)	9.3 (15.2)	17.2 (27.3)	12.1 (19.3)	22.5 (35.6)
2	V_{ext}	PBE	6.0 (9.7)	9.0 (14.7)	19.5 (32.8)	10.0 (15.4)	21.3 (34.0)
3	V_{ext}	PBE	10.9 (15.3)	16.8 (23.8)	46.1 (69.5)	16.3 (24.2)	43.6 (65.4)
10	V_{ext}	PBE	7.1 (10.6)	10.8 (16.1)	49.3 (85.4)	10.4 (16.8)	39.4 (71.5)
1	V_{ext}	MGGA	5.7 (9.8)	12.0 (20.4)	16.1 (25.3)	10.9 (17.9)	21.7 (34.8)
2	V_{ext}	MGGA	5.4 (8.6)	11.0 (17.8)	14.8 (23.2)	8.1 (12.4)	19.3 (30.0)
3	V_{ext}	MGGA	11.1 (15.5)	22.9 (32.3)	44.5 (67.9)	16.7 (25.0)	43.0 (65.7)
10	V_{ext}	MGGA	7.3 (10.8)	15.0 (22.2)	50.8 (88.0)	10.8 (16.9)	41.4 (76.0)
1	ρ	LDA→PBE	1.2 (2.0)	1.7 (3.0)	14.5 (31.1)	2.4 (3.6)	14.6 (30.5)
2	ρ	LDA→PBE	1.2 (4.3)	1.8 (6.4)	8.6 (19.3)	1.9 (4.6)	9.1 (23.9)
3	ρ	LDA→PBE	1.9 (3.4)	2.8 (5.0)	34.8 (56.2)	5.1 (7.7)	35.6 (56.9)
10	ρ	LDA→PBE	2.2 (4.0)	3.2 (6.1)	75.5 (118.4)	7.6 (11.7)	76.7 (120.9)
1	ρ	LDA→MGGA	1.4 (2.8)	2.9 (5.8)	14.2 (30.9)	2.6 (4.5)	14.7 (31.6)
2	ρ	LDA→MGGA	1.5 (3.7)	3.0 (7.5)	8.0 (15.2)	2.0 (4.3)	8.7 (19.6)
3	ρ	LDA→MGGA	2.9 (5.4)	6.0 (11.3)	36.3 (58.2)	5.8 (10.0)	37.7 (59.8)
10	ρ	LDA→MGGA	2.6 (4.9)	5.3 (9.9)	73.6 (115.6)	7.5 (11.7)	74.4 (117.4)
10	$V_{\text{ext}} \to \rho$	LDA	6.4 (11.0)	9.9 (17.1)	93.0 (151.4)	12.7 (21.5)	98.9 (167.5)
10	$V_{\text{ext}} \to \rho$ (DDE)	LDA	6.4 (10.7)	10.0 (16.7)	99.4 (154.4)	13.4 (22.2)	108.2 (175.5)
10	Predicted ρ	LDA	5.8 (10.1)	9.0 (15.7)	91.4 (143.2)	11.2 (19.1)	98.5 (158.7)
10	$V_{\text{ext}} \to \rho$	PBE	7.2 (11.7)	10.9 (17.7)	100.4 (164.8)	14.5 (24.6)	108.1 (185.8)
10	$V_{\text{ext}} \to \rho$ (DDE)	PBE	7.2 (11.5)	10.9 (17.7)	107.0 (168.1)	15.1 (24.9)	117.6 (193.7)
10	Predicted ρ	PBE	6.8 (11.1)	10.2 (16.8)	98.2 (151.5)	12.4 (21.4)	106.7 (170.3)
10	$V_{\text{ext}} \to \rho$	MGGA	7.5 (12.3)	15.4 (25.1)	94.6 (161.9)	13.0 (22.8)	103.3 (180.5)
10	$V_{\text{ext}} \to \rho$ (DDE)	MGGA	7.4 (12.0)	15.2 (24.6)	101.1 (162.4)	13.7 (23.3)	111.8 (184.0)
10	Predicted ρ	MGGA	7.1 (12.1)	14.7 (24.8)	94.9 (150.2)	12.7 (21.5)	102.5 (167.3)

Table 8.3: Mean absolute errors (in mHa per electron) and root mean squared errors (in parenthesis) for models trained in this report for the RND potentials. The abbreviations ρ, V_{ext}, and Predicted ρ are charge density, potential, and predicted charge density respectively. The arrows (i.e. LDA→PBE) indicate that the charge density used as input to the DNN was calculated using the LDA exchange-correlation functional, but the labels (energies) were calculated using another exchange-correlation functional. The acronym DDE stands for density driven error, as defined by Brockherde *et al.* [17]. The models labelled by $V_{\text{ext}} \to \rho$ directly map the external potentials to charge densities. The models labelled by Predicted ρ map predicted charge densities to true energies.

8.A.4 A note on the density driven errors

In Table 8.3, we report the mean absolute density driven error rather than the density driven error that is reported in [17]. When evaluating a machine learning model, it is common to report absolute errors to avoid reporting an average error that would have error cancellation. Consider the total energy expression

$$E[\rho] = F[\rho] + \int d\mathbf{r}\, V(\mathbf{r})\rho(\mathbf{r}) \tag{8.4}$$

where F is the universal functional defined in [41] and V is the external potential. The value E is the *true* energy given a *true* charge density ρ. The total error reported in [17] is defined to be

$$\Delta E = \tilde{E}[\tilde{\rho}] - E[\rho] = \Delta E_F + \Delta E_D \tag{8.5}$$

where

$$\Delta E_F = \tilde{F}[\rho] - F[\rho] \tag{8.6}$$

is the functional driven error and

$$\Delta E_D = \tilde{E}[\tilde{\rho}] - \tilde{E}[\rho] \tag{8.7}$$

is the density driven error. The variables $\tilde{\rho}$, $\tilde{E}$, and $\tilde{F}$ represent a predicted charge density, an approximation to the total energy functional, and an approximation to true universal functional, respectively. If we consider the absolute value of the total error

$$|\Delta E| = |\Delta E_F + \Delta E_D| \neq |\Delta E_F| + |\Delta E_D| \tag{8.8}$$

then we can see that there must be error cancellation between the terms $\Delta E_F + \Delta E_D$ in order for $|\Delta E| < |\Delta E_D|$. This is what we find in Table 8.3.

Chapter 9

Orbital-Free Density Functional Theory with Small Datasets and Deep Learning

In Chapter 8, we showed that extensive deep neural networks (EDNNs) can be used to predict properties calculated in density functional theory (DFT). EDNNs are inherently extensive and have the capacity to scale up to arbitrary system size, allowing for large scale inference. However, one still must generate large datasets to train them, which in itself can be very computationally demanding. In the following Chapter, we aim to solve this "data problem." We use voxel deep neural networks (VDNNs) to map scalar fields with an image-based approach. In addition, VDNNs are also extensive, and allow one to simulate arbitrary system sizes. When one considers predicting a field rather than a scalar value, one quickly finds themselves in a regime where data is plentiful, and must sample the data to avoid bias in the machine learning (ML) model. However, VDNNs don't solve the transferability problem. As shown in Chapter 7, ML models do not extrapolate well, and one must consider different areas of chemical space so that the model can traverse through it without an increase in error. This is especially important when searching for new materials within a design pipeline. Normally, in quantum chemistry, one considers structures with different bonds or conformations, such that the energy landscape is a large and diverse one. With VDNNs, the focus is a density for both inputs and outputs, and one must consider the scalar function that is being input into the model when trying to create a diverse training set. If a new environment is comparable to one previously seen, the VDNN will produce accurate results.

This work is currently under review.

Kevin Ryczko
Department of Physics, University of Ottawa, Ottawa, Ontario, Canada
Vector Institute for Artificial Intelligence, Toronto, Ontario, Canada

Sebastian J. Wetzel
Perimeter Institute for Theoretical Physics, Waterloo, Ontario, Canada

Roger G. Melko
Perimeter Institute for Theoretical Physics, Waterloo, Ontario, Canada
Department of Physics and Astronomy, University of Waterloo, Waterloo, Ontario, Canada

Isaac Tamblyn
Department of Physics, University of Ottawa, Ottawa, Ontario, Canada
Vector Institute for Artificial Intelligence, Toronto, Ontario, Canada

Abstract

We use voxel deep neural networks to predict energy densities and functional derivatives of electron kinetic energies for the Thomas-Fermi model and Kohn-Sham density functional theory calculations. We show that the ground-state electron density can be found via direct minimization for a graphene lattice without any projection scheme using a voxel deep neural network trained with the Thomas-Fermi model. Additionally, we predict the kinetic energy of a graphene lattice within chemical accuracy after training from only 2 Kohn-Sham density functional theory calculations. Furthermore, we demonstrate an alternative, functional derivative-free, Monte Carlo based orbital free density functional theory algorithm to calculate an accurate 2-electron density in a double inverted Gaussian potential with a machine-learned kinetic energy functional.

9.1 Introduction

Kohn-Sham density-functional theory [41] (KS-DFT) and Orbital-Free (OF) DFT [124, 125] are two electronic structure methodologies to calculate properties of matter. In OF-DFT, all energy functionals depend only on the electron density, whereas in KS-DFT, energy functionals depend on both the non-interacting electron density and the set of Kohn-Sham orbitals. The explicit dependence on the electron density in OF-DFT allows for favourable, $\mathcal{O}(N)$, computational scaling, enabling one to study large systems [126] (where N is the number of electrons). Conversely, The computational scaling of KS-DFT ($\mathcal{O}(N^3)$) is less favourable due to the computation of a set of orbitals, rather than the electron density alone. However, the main advantage of KS-DFT implementations is that the kinetic energy is calculated via a single-particle quantum mechanical operator, leading to a more accurate approximation of the true kinetic energy functional (KEF) compared to OF-DFT. In OF-DFT, the kinetic energy is written as an approximate functional of the electron density. The lack of knowledge of the true, quantum mechanical KEF reduces the accuracy and applicability of OF-DFT.

Thomas and Fermi (TF) both proposed an analytic KEF assuming a free electron gas [38, 39]. They were followed by the Thomas-Fermi-Dirac-von Weizsäcker and $X\alpha$ models [127, 128, 129, 130] to address the failures of the TF model for atoms and molecules. Hohenberg and Kohn [40] proved the existence of a KEF that depends explicitly on the electron density of interacting electrons, but never gave its exact functional form. Subsequently, Kohn and Sham [41] introduced a non-interacting, orbital-dependant KEF. This non-interacting functional is routinely used in all KS-DFT calculations.

More recently, machine learning models have been used as energy functionals [14, 17, 131, 132, 133, 5]. Specifically, in Refs. [14, 131] machine-learned, one-dimensional KEFs were constructed using kernel ridge regression and convolutional neural networks (CNNs). In Ref. [14], the authors argued that the error of a functional derivative of a machine-learned KEF (FD-KEF) was too large to be used in a direct minimization calculation. They reduced

this error by projecting the functional derivative of the total energy onto a subspace found with principal component analysis. Following this report, Ref. [131] included the FD-KEF in a loss function to improve the predictions from the machine learning models. This improved loss function reduced the prediction error of the FD-KEF but did not eliminate it entirely. An additional projection method using a sinusoidal basis was introduced and utilized to minimize the error. The use of a sinusoidal basis eliminated the computational overhead of performing principal component analysis on the training set densities.

In addition to KEFs, machine learning models have been used as exchange-correlation functionals [30, 134, 135]. In Refs. [30, 134], “slices" of the density, rather than the entire scalar field, were used as input to neural networks. It was shown that machine-learned exchange-correlation functionals could be used for a model system with a simple harmonic oscillator potential, several molecules, and a unit cell of Si, demonstrating the transferability of this methodology. Additionally, the approach drastically reduced the number of calculations needed to generate a training set.

We build on previous work which computed KEFs for one-dimensional systems and compute KEFs, FD-KEFs, electron densities, and energies in *three dimensions for a realistic system*: pristine graphene lattices. We also *eliminate the need for large datasets*. Namely, we use slices of the electron density as input to deep neural networks (DNNs), where the output is also a slice of the kinetic energy density (KED). Desired quantities are subsequently found via integration over the supercell. We call this methodology voxel DNNs (VDNNs). In Section 9.2, we outline the basic electronic structure, training data generation, and machine learning methodologies used. In Section 9.3, we outline the results of VDNNs in practice. We first investigate the Thomas-Fermi model with VDNNs as a proof of principle. The Thomas-Fermi model is simple and both the kinetic energy and its functional derivative with respect to the electron density are analytically known for all densities. Afterwards, we apply VDNNs to KS-DFT. Using VDNNs allows one to have a Kohn-Sham kinetic energy functional that explicitly depends on the electron density and enables one to insert the energy functional into OF-DFT (Figure 9.1). Lastly, we show an alternative potential of

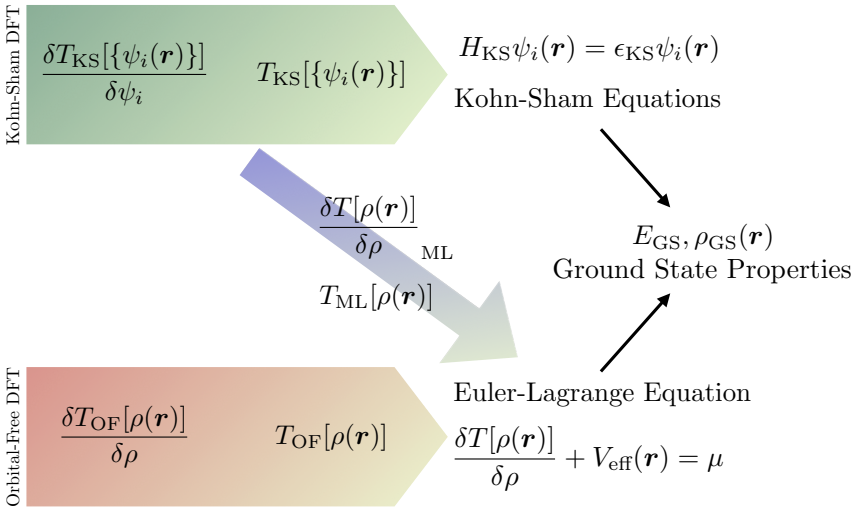


Figure 9.1: Our machine learning architecture, similar to Refs. [14, 131], makes a connection between Kohn-Sham density functional theory and orbital-free density functional theory. The model allows for the construction of Kohn-Sham kinetic energy functionals that explicitly depend on the electron density and, therefore, direct insertion into orbital-free density functional theory. See Section 9.2 for more information about the equations.

our method with a demonstration of a functional derivative-free Monte Carlo (MC) based optimization for a toy, 1D model system. Direct minimization techniques have been applied in KS-DFT calculations [136, 137] which avoids the self-consistent procedure, but direct minimization in OF-DFT still requires functional derivatives. Our MC based optimization eliminates the need of a functional derivative altogether. We conclude and propose future directions based on our results in Section 9.4.

9.2 Methods

In this work, we use VDNNs to calculate KEDs ($\mathcal{T}$) and FD-KEFs ($\mathcal{F}$) of graphene lattices using OF-DFT with the Thomas-Fermi model and using KS-DFT (LDA and GGA). As discussed above, the Thomas-Fermi model serves as a preliminary experiment due to its simplicity and KS-DFT serves as a realistic use case. We therefore first test our methodology with the Thomas-Fermi model before moving on to KS-DFT. In OF-DFT, the total energy

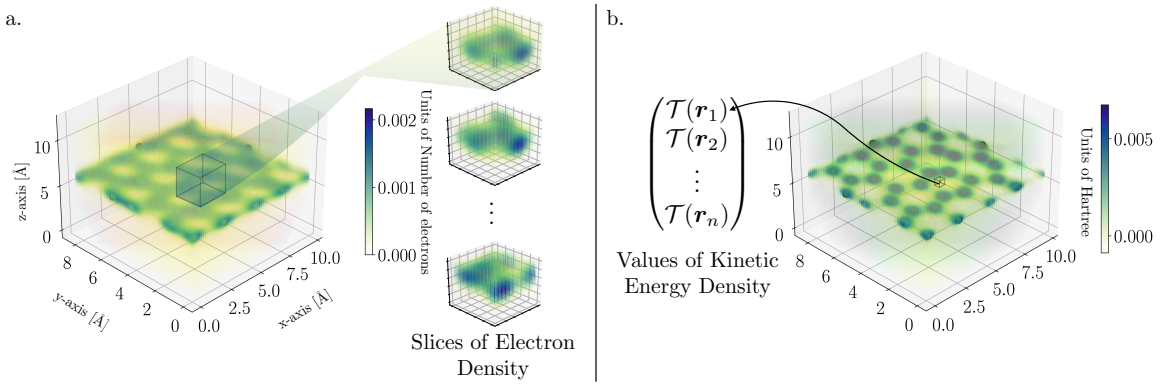


Figure 9.2: A visual representation of voxel deep neural networks. (a) An example electron density for a 32 atom graphene lattice supercell. The highlighted region in the electron density is a slice of the electron density centered at a particular pixel. (b) The kinetic energy density for the same 32 atom graphene lattice. The voxel deep neural network learns the mapping between the slice of electron density to the voxel of kinetic energy density.

functional is written in real space as

$$E[\rho(\boldsymbol{r})] = T[\rho(\boldsymbol{r})] + E_{\text{Hartree}}[\rho(\boldsymbol{r})] + E_{\text{ion}}[\rho(\boldsymbol{r})] + E_{\text{xc}}[\rho(\boldsymbol{r})] \tag{9.1}$$

where $\rho(\boldsymbol{r})$ is the electron density and the terms in order are kinetic, Hartree, external, and exchange-correlation energies. To find the ground state electron density, one searches for an electron density which minimizes the total energy expression under the constraint that the number of electrons, N_e, is fixed. This yields the Lagrangian

$$\mathcal{L}[\rho(\boldsymbol{r})] = E[\rho(\boldsymbol{r})] - \mu\left(\int_{\Omega} d\boldsymbol{r}\ \rho(\boldsymbol{r}) - N_e\right) \tag{9.2}$$

where μ is a Lagrange multiplier and Ω is the volume of the supercell. Applying a functional derivative of the Lagrangian with respect to the electron density yields the Euler-Lagrange equation

$$\mathcal{F}(\boldsymbol{r}) + V_{\text{eff}}(\boldsymbol{r}) = \mu, \tag{9.3}$$

where

$$V_{\text{eff}}(\boldsymbol{r}) = V_{\text{Hartree}}(\boldsymbol{r}) + V_{\text{ion}}(\boldsymbol{r}) + V_{\text{xc}}(\boldsymbol{r}), \tag{9.4}$$

and

$$\mathcal{F}(\boldsymbol{r}) = \frac{\delta T[\rho(\boldsymbol{r}')](\boldsymbol{r})}{\delta \rho(\boldsymbol{r}')}. \tag{9.5}$$

Using gradient descent, one can solve for the ground state electron density via direct minimization

$$\phi_{n+1}(\boldsymbol{r}) = \phi_n(\boldsymbol{r}) - 2\alpha\phi_n(\boldsymbol{r})\left(\mathcal{F}(\boldsymbol{r}) + V_{\text{eff}}(\boldsymbol{r}) - \mu\right)_n \tag{9.6}$$

where $\phi_n(\boldsymbol{r}) = \sqrt{\rho_n(\boldsymbol{r})}$, and α is a small parameter. The use of the square root of the density ensures that the electron density remains positive during the optimization.

Using the Thomas-Fermi model and the DFTpy code [138], we performed 2 direct minimization calculations for 32-atom slabs of graphene where the atoms were perturbed from their equilibrium geometry. The perturbations were generated from a normal distribution with a standard deviation of 0.1 Å. We used an energy cutoff of 45 Ha, the LDA exchange-correlation functional [41], and norm-conserving pseudopotentials [139]. Due to the free electron gas approximation used for the kinetic energy, we maintained this approximation in our exchange-correlation functional choice. We collected ρ_{TF}, $\mathcal{T}_{\text{TF}}$, and $\mathcal{F}_{\text{TF}}$ every 10 steps (values of n in Equation 9.6) from one of the calculations to be used as training data. This made for a total of 173 training configurations. The second calculation was used as independent test data.

In addition to OF-DFT calculations, we used KS-DFT to investigate 32-atoms graphene slabs where the atoms were perturbed in the same way as described above. In KS-DFT, the electron density is written as

$$\rho_{\text{KS}}(\boldsymbol{r}) = 2\sum_{n}^{\text{occ}}\sum_{\boldsymbol{k}} w_{\boldsymbol{k}}\psi^*_{n,\boldsymbol{k}}(\boldsymbol{r})\psi_{n,\boldsymbol{k}}(\boldsymbol{r}) \tag{9.7}$$

and the KED is written as

$$\mathcal{T}_{\text{KS}}(\boldsymbol{r}) = -\sum_{n}^{\text{occ}}\sum_{\boldsymbol{k}} w_{\boldsymbol{k}}\psi^*_{n,\boldsymbol{k}}(\boldsymbol{r})\nabla^2\psi_{n,\boldsymbol{k}}(\boldsymbol{r}). \tag{9.8}$$

In Equations 9.7 and 9.8, n is the band index, $\boldsymbol{k}$ is the k-point, $w_{\boldsymbol{k}}$ is the weighting associated with each k-point and ψ is a Kohn-Sham orbital. In this work, we use finite differences to compute derivatives of the Kohn-Sham orbitals. To compute the Kohn-Sham orbitals we used Abinit [140] with an energy cutoff of 45 Ha, a $4 \times 4 \times 1$ k-point grid, the PBE exchange-correlation functional [44] and norm-conserving pseudopotentials [139]. We justify this exchange-correlation choice based on its popularity in the literature. Here, we performed a total of 200 DFT calculations where 100 of the configurations were for training and 100 for kept aside for testing. To obtain $\mathcal{F}_{\text{KS}}$ for these calculations, we used Equation 9.3 where the potentials were evaluated using DFTpy [138], and the chemical potentials were obtained from Abinit. It should be noted that Equation 9.3 can only be used to define $\mathcal{F}_{\text{KS}}$ when self-consistency has been reached [141].

To train the VDNNs, we collected slices of ρ (and $\nabla\rho$ for Kohn-Sham models) as inputs and slices of $\mathcal{T}$ and $\mathcal{F}$ as outputs. If $\widetilde{\mathcal{T}}$ and $\widetilde{\rho}$ are discretized forms of $\mathcal{T}$ and ρ then a slice of ρ with dimensions (a, b, c) centred at pixels (i, j, k) is written as $\widetilde{\rho}[i - a/2 : i + a/2 + 1, j - b/2 : j + b/2 + 1, k - c/2 : k + c/2 + 1]$. The addition of 1 is due to the use of odd values of a, b, c. A slice of $\mathcal{T}$ with dimensions (a', b', c') centred at pixels (i, j, k) is similarly written as $\widetilde{\mathcal{T}}[i - a'/2 : i + a'/2 + 1, j - b'/2 : j + b'/2 + 1, k - c'/2 : k + c'/2 + 1]$. We tested a variety of input sizes and used output sizes of (1,1,1). This corresponds to mapping electron density slices to values of $\mathcal{T}$, as shown in Figure 9.2. To avoid bias in training, we sample $\mathcal{T}$ or $\mathcal{F}$ such that a uniform distribution is produced given a target number of samples. The target number of samples was 1024^2 unless stated otherwise. Of these, 99% of them were used for training, and 1% were used for validation. Testing was done on the 100 independent DFT calculations not seen during training. Inputs were standardized and normalized such that the range of values was $\in [-1, 1]$ and outputs were normalized $\in [0, 1]$. We used a modified version of the deep neural network (DNN) architecture used in Refs. [142, 143], which had success in predicting various energies at the DFT level with different functionals. This included 2 non-reducing convolutional layers with 64 $3 \times 3 \times 3$ kernels, 4 non-reducing convolutional layers with 16 $3 \times 3 \times 3$ kernels, a reducing convolutional layer with 64 $3 \times 3 \times 3$ kernels, 4 non-reducing convolutional layers with 32 $3 \times 3 \times 3$ kernels, a fully connected

layer with 1024 neurons, and a fully connected layer with 2 outputs. Since the inputs are scalar fields, a natural architectural choice is to use convolutional layers. They are designed to extract relevant features from images to make accurate predictions. The input dimensions are less than in Refs. [142, 143], which is why the first 2 convolutional layers were changed to non-reducing layers. It has been found previously that the ELU activation function has improved results compared to RELU with batch normalization [144]. We note that this particular architecture choice is most likely not optimal, and one could obtain better results with another architecture choice. Models were trained for 500 epochs with learning rates of 10^{-5} and a batch size of 512. Production models were trained across 16 NVIDIA V100 GPUs with layer-wise adaptive rate scaling with clipping [56]. Training on large batch sizes leads to unfavourable results and Ref. [56] have shown that layer-wise adaptive rate scaling allows one to obtain similar results to lower batch training while reducing the training time. Inference for the densities can be trivially parallelized and does not suffer from any negative large-batch effects. It was done across 64 NVIDIA V100 GPUs. Our method does not require this GPU setup, but can make use of them when performing inference on large grids. Our multi-node, multi-GPU training code and our multi-node, multi-GPU inference code can be found here [145].

9.3 Results

We first discuss using VDNNs for the TF model. After training on $\mathcal{T}_{\text{TF}}$ and $\mathcal{F}_{\text{TF}}$ simultaneously, where $\mathcal{F}_{\text{TF}}$ was uniformly sampled, we study the accuracy of the model on the validation and testing data. Looking to Figure 9.3a-b, we plot residuals for $\mathcal{T}_{\text{TF}}$ and $\mathcal{F}_{\text{TF}}$ for the validation set in units of meV and meV / electron, respectively. Density values have been multiplied by the volume such that direct integration over the numerical grid yields units of energy or energy / electron. From these plots, we can see that differences are a small fraction of the energy values. MAEs for $\mathcal{T}_{\text{TF}}$ and $\mathcal{F}_{\text{TF}}$ are 0.04 meV, and 0.08 meV / electron. RMSEs for $\mathcal{T}_{\text{TF}}$ and $\mathcal{F}_{\text{TF}}$ are 0.05 meV, and 0.11 meV / electron. The error for $\mathcal{F}_{\text{TF}}$ is larger than $\mathcal{T}_{\text{TF}}$. Part of this increase in error can be attributed to the increase in the range of

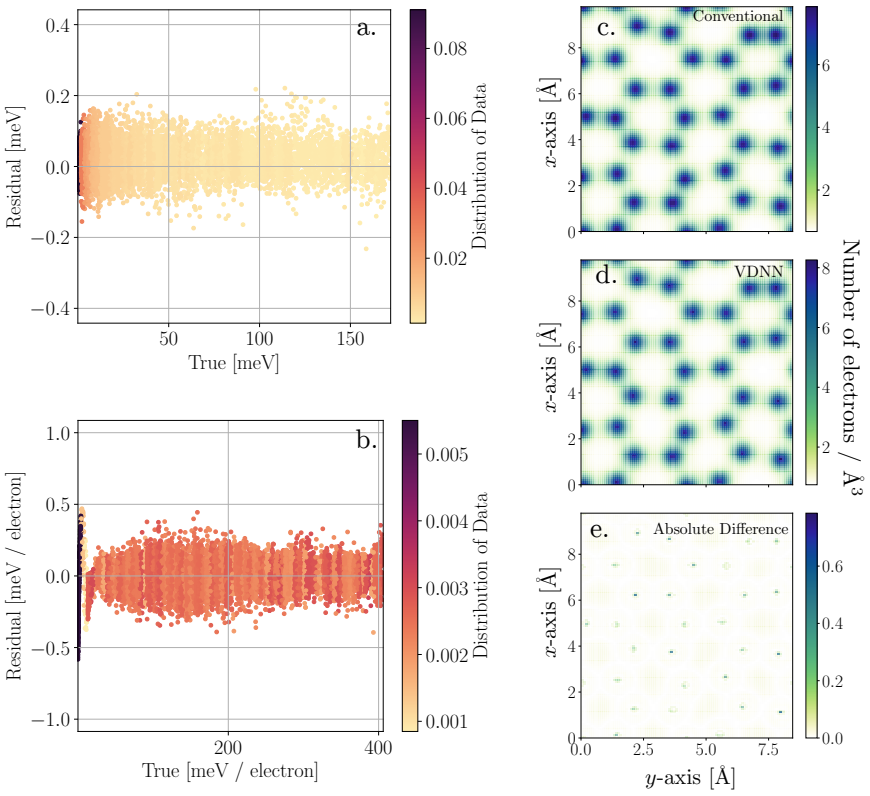


Figure 9.3: Thomas-Fermi model: Residual (true minus predicted) versus true for (a) $\mathcal{T}_{\text{TF}}$ and (b) $\mathcal{F}_{\text{TF}}$. (c) Thomas-Fermi electron density from a traditional direct minimization calculation and (d) electron density from a direct minimization calculation with a VDNN trained from a single OF-DFT calculation. (e) Absolute differences between the densities. VDNNs can be used in direct minimization calculations to find electron densities for the Thomas-Fermi model.

values (a factor of 2.67 from $\mathcal{T}_{\text{TF}}$ to $\mathcal{F}_{\text{TF}}$), which contributes to 96% of the increased error; the remaining increase in error is due to the VDNN.

We now use VDNNs to calculate an electron density and energy for the second, testing configuration via Equation Equation 9.3. In past reports [14, 29], it was declared unfeasible to directly solve Equation 9.3 because the derivatives of the machine learning model had too much noise. In Ref. [14] noise was reduced by projecting functional derivatives onto a subspace spanned by relevant vectors via principal component analysis. A similar approach was taken in Ref. [29], where they projected the functional derivatives onto a subspace spanned by a sinusoidal basis. Here, without any projection scheme, we show that it is possible to use Equation 9.6 to solve for an electron density directly. A projection scheme is not

necessary since no derivatives are being taken with respect to the DNN. The VDNN directly outputs the kinetic energy and the functional derivative of the kinetic energy. We used a value of $\alpha = 10^{-3}$ and a uniform electron density as the starting guess. We re-normalized the electron density at every step to enforce charge conservation and deemed a calculation converged when the absolute change of the energy between subsequent steps was less than 10^{-4} Ha. The exact electron density and the electron density found using the VDNN are shown in Figure 9.3. The densities differ minimally, and the total energy difference found between the two calculations was 19.1 meV / electron. Thus, machine learning models *can* be used in direct minimization calculations without any sort of projection scheme for the Thomas-Fermi model.

We now consider $\mathcal{T}_{\text{KS}}$ and $\mathcal{F}_{\text{KS}}$. After generating a training dataset that uniformly sampled $\sqrt{\rho}\mathcal{F}_{\text{KS}}$, we trained a VDNN on $\mathcal{T}_{\text{KS}}$ and $\sqrt{\rho}\mathcal{F}_{\text{KS}}$ simultaneously with ρ, $\partial_x\rho$, and $\partial_y\rho$ as input. We found that including gradients as input channels reduced the mean squared error on the validation set by 7%. Training on $\sqrt{\rho}\mathcal{F}_{\text{KS}}$ rather than $\mathcal{F}_{\text{KS}}$ reduced the mean squared error on the validation set by 43%. Multiplication of $\sqrt{\rho}$ eliminates $\mathcal{F}_{\text{KS}}$ where $\rho = 0$, and enhances $\mathcal{F}_{\text{KS}}$ where $\rho \neq 0$. This filter-like behaviour allows for an improvement in the predictions. In Figure 9.4a, we plot the true energy per electron versus residual energy per electron for the 100 testing atomic configurations. To determine percentage errors, the mean of the true kinetic energy values was used. From here, we see that all predictions are within chemical accuracy (43.4 meV). The MAE and RMSE were 4.3 meV / electron and 5.6 meV / electron respectively. In Figure 9.4b, we plot true versus residual values for $\int_\Omega d\boldsymbol{r}\ \sqrt{\rho(\boldsymbol{r})}\mathcal{F}_{\text{KS}}(\boldsymbol{r})$. From here we find that all values are well within 0.25% error. MAE and the RMSE were 0.67 meV / electron$^{3/2}$ and 0.83 meV / electron$^{3/2}$. VDNNs can provide all of the relevant information needed in OF-DFT. However, using Equation 9.6, we were unable to obtain the correct electron density for the Kohn-Sham models. This failure is not due to errors of the model, but the lack of knowledge of $\mathcal{F}_{\text{KS}}$ for *unconverged* electron densities. In previous work [14, 29], and for the KS-DFT data, functional derivatives of the kinetic energy are collected for only converged calculations. When using Equation 9.6, one encounters unconverged electron densities, and must also know the mapping from these

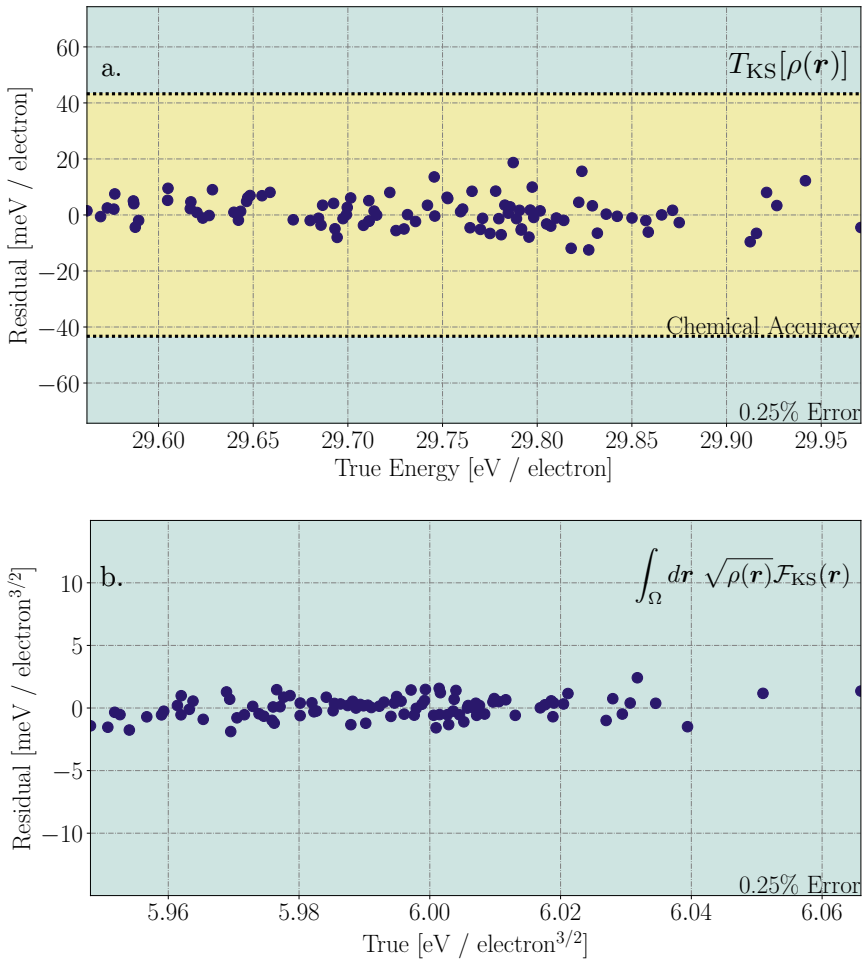


Figure 9.4: Residuals of (a) $T_{KS}[\rho(\boldsymbol{r})]$ and (b) $\int_\Omega d\boldsymbol{r}\ \sqrt{\rho(\boldsymbol{r})}\mathcal{F}_{KS}(\boldsymbol{r})$ for the Kohn-Sham model. Predictions are for a test set containing 100 graphene systems with 32 atoms as described in Section 9.2. VDNNs allow for accurate predictions from small Kohn-Sham density functional theory datasets.

unconverged electron densities to their respective kinetic energy densities and functional derivatives of the kinetic energies. Although $\mathcal{T}_{KS}$ is known for all iterations, $\mathcal{F}_{KS}$ is not. This lack of knowledge prevents the insertion of more accurate, kinetic energy machine learning frameworks in OF-DFT. This highlights the need for future work in this area. Solving this challenge would significantly reduce the amount of computation for accurate electronic structure calculations. As shown recently in Ref. [146], this problem could be solved by considering a differential equation that includes $\mathcal{F}$ and a source function. This source function depends explicitly on the electron density, and $\mathcal{F}$ can be found once this source function is known. Unfortunately, for KS-DFT calculations this source function is also only

known for converged electron densities, but further work in this area could be promising.

We also investigated how VDNNs perform on a toy, 2 electron system in 1D previously investigated in Refs. [14, 131]. We used the same ResNet architecture and dataset as described in Ref. [29], with a field of view of 257 voxels for the VDNN and we compare our results to the ResNet model of Ref. [29]. We found that using $\sqrt{\rho}\mathcal{F}$ also yielded a smaller mean squared error during training, as described above for $\mathcal{F}_{\text{KS}}$. For T, our error was ≈ 75 times larger than Ref. [29] with a MAE of 0.17 eV (3.84 kcal / mol). This large discrepancy is due to the previous models being trained directly on the energy rather than energy density. This allowed for highly accurate models with errors an order of magnitude less than chemical accuracy. For $\mathcal{F}$, we found that our error was ≈ 1.9 times larger with a MAE of 0.50 eV / electron (11.42 kcal / mol / electron). However, for $\mathcal{F}$, our maximum absolute error was 1.8 times smaller. In addition, when comparing the errors between T for the 1D system and the 3D system (T_{KS}) we find an increase in error by a factor of ≈ 20 for the 1D system. As we change the number of physical dimensions, the number of inputs to the model increases. The number of pixels in the 3D case (19^3) is ≈ 20 times larger compared than the 1D case (257). Networks have more information to extract features from, which leads to more accurate predictions. It should also be noted that the VDNN is capable of performing inference for an arbitrarily sized 1D system, so long as the potentials and electron densities are similar to the training set. The existing models from Refs. [14, 131] are limited to the same system sizes used during training.

An alternative approach to minimizing Equation 9.1 that avoids computing functional derivatives is MC optimization via the Metropolis algorithm [54]. Direct minimization approaches often require information about derivatives to make a gradient based update. Gradient free optimization is an alternative approach that does not require such information and is more compatible with machine learning methods since the computational cost associated with inference is low and derivatives can be unreliable. To showcase this potential solution, we consider 2 electrons in 1 dimension with the Thomas-Fermi model as the kinetic energy functional. Using this approximation allows us to compare our MC based

optimization with a traditional, gradient based optimization. The total energy functional, excluding exchange-correlation effects, can be written as

$$\begin{aligned} E[\rho] &= \frac{\pi^2}{12}\int_\ell dx\, \rho^3(x) + \frac{1}{2}\int_\ell dx \int_\ell dx' \frac{\rho(x')\rho(x)}{|x-x'|} \\ &+ \int_\ell dx \sum_{i=1}^{2} -\alpha_i \exp(-(x-\beta_i)^2)\rho(x) \end{aligned} \tag{9.9}$$

where ℓ is the length of the 1 dimensional cell. In Section 9.3, the first term is the kinetic energy of the 1 dimensional Thomas-Fermi model, the second term is the 1 dimensional Hartree energy, and the third term is the external energy from a toy, double inverted Gaussian potential. For the external energy, we used the parameters: $\alpha_1 = 1.0$ Ha / electron, $\alpha_1 = 2.0$ Ha / electron, $\beta_1 = -0.5$ Bohr, $\beta_2 = 1.0$ Bohr. For the kinetic energy, we trained a 3 layer, fully connected neural network that maps a value of ρ to a value of the one dimensional kinetic energy density. We generated 10^5 random numbers from 0 to 1, which represented values of density, and trained the network for 100 epochs with a batch size of 100 and a learning rate of 10^{-5}. We did not perform any standardization or normalization and we used ELU activation functions throughout the network. For the MC simulation, we performed simulated annealing with a starting value of $\beta^{-1} = 10^{-4}$ Ha which was decreased according to the formula $\beta_{\text{new}}^{-1} = \beta_{\text{old}}^{-1}/(1.0 + 2 \times 10^{-6})^n$, where n is the iteration number. After 2 million iterations, $\beta^{-1} = 1.87 \times 10^{-6}$ Ha. At each iteration, we updated all values of ρ in two steps. The first step was computing a random change

$$\Delta\rho = 1000(\rho(x) + 10\sigma)u(\sigma, x) \tag{9.10}$$

where σ is the standard deviation and $u(\sigma, x)$ is function generated from a normal distribution centered at zero with the same shape as $\rho(x)$. The random change is then updated according to

$$\Delta\rho = \Delta\rho - \rho\frac{\langle\Delta\rho\rangle}{\langle\rho\rangle} \tag{9.11}$$

where $\langle f\rangle$ is the mean of f. We used a standard deviation of $\sigma = 10^{-5}$ which was reduced during the simulation following the same protocol as β. All proposed values of ρ that were negative were set to zero, and ρ was re-normalized at every step before evaluating

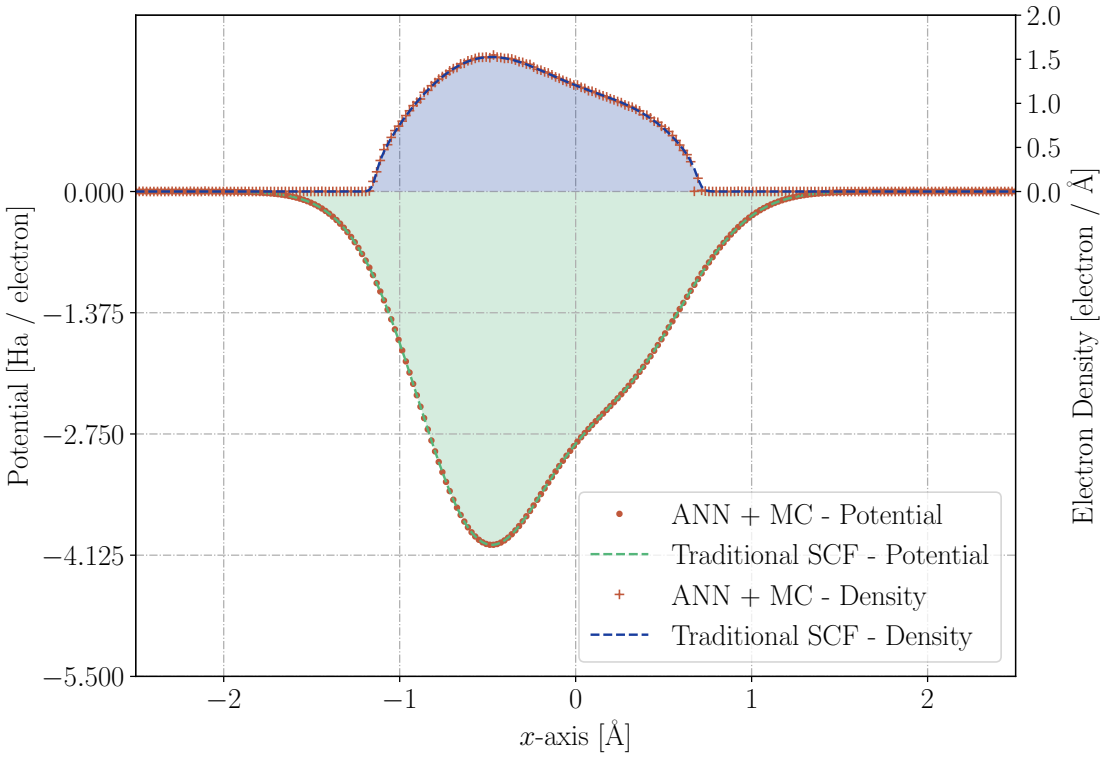


Figure 9.5: Comparison of electron densities and potentials between a Monte Carlo optimization (red crosses and points) and a traditional self-consistent field calculation (blue and green dashed lines) for 2 electrons in 1D with the external potential described in Section 9.3. The Monte Carlo optimization yields an electron density that is very similar to a self-consistent field calculation.

the energy. In Figure 9.5, we plot ρ and the potential (Hartree + external) for a traditional direct minimization calculation, following Equation 9.6 alongside ρ and the potential for the MC simulation. For the traditional gradient based calculation, the energy was declared converged when the difference in energy between subsequent steps was $< 10^{-6}$ Ha. There is excellent agreement between the MC optimization and the gradient based optimization. The mean absolute differences between the charge densities, potentials, and total energies were 3.74×10^{-3} electron / Å, 5.61×10^{-5} meV / electron, and 1.70 meV respectively. Future work involves implementing a 3 dimensional, functional derivative-free, OF-MC algorithm capable of calculating more accurate electron densities with improved, machine-learned kinetic energy functionals.

9.4 Conclusion

We have shown that VDNNs can be used to accurately predict the kinetic energy density and the functional derivative of the kinetic energy for Kohn-Sham and Thomas-Fermi theories. This methodology drastically reduces the number of electronic structure calculations needed to generate a training set. We have shown that one can obtain an accurate charge density and total energy after training with data from only 1 direct minimization calculation for the Thomas-Fermi model. Similarly, we have shown that we can calculate accurate kinetic energies from only 2 converged calculations for Kohn-Sham density functional theory. Additionally, we show that this accuracy is held to arbitrary system size. Currently, one cannot use voxel deep neural networks to represent the functional derivative of the kinetic energy from Kohn-Sham because an expression does not currently exist for densities that do not satisfy the Euler equation. However, we show that if such an expression exists, one could use a voxel deep neural network in a self-consistent calculation to solve for an electron density. An alternative, functional derivative-free, Monte Carlo based orbital-free algorithm could also be used to determine ground state electron densities.

9.5 Acknowledgements

The authors acknowledge fruitful discussion with Pierre Darancet and for comments on the manuscript. The authors also acknowledge Compute Canada, the National Research Council, and the Vector Institute for Artificial Intelligence for computational resources. KR acknowledges the National Sciences and Engineering Council of Canada, and the Vector Institute for Artificial Intelligence for funding. Work at the National Research Council was carried out under the auspicious of the AI4D program.

9.A Supplemental Information

9.A.1 Hyperparameter Studies and Additional Results

Before using VDNNs in practice, we focus on determining hyper-parameters using $\mathcal{T}_{\text{KS}}$.

To answer the question of optimal input size, we trained VDNNs with different input sizes and compared errors of different models. In Figure 9.6a, we show the normalized mean squared error of the validation sets as a function of input size. The length of inputs in each dimension is the same. For example, the input size of 19 corresponds to an input image with dimensions 19^3. From Figure 9.6a, we see that as the image size is increased, the error decreases. We also see that the error is converging; beyond a certain input size, the addition of extra pixels is not advantageous. As we increase the input size, the training and inference computational cost also increase. This can also be seen in Figure 9.6a, where we plot the average epoch time as a function of input size. In this case, the computational cost increases linearly with the number of pixels. Thus when one chooses an input size, there is a balance between accuracy and computational cost. We found input sizes of 19^3 were a good trade-off between accuracy and computational cost.

How many input examples are needed to produce an accurate model? To answer this question, we trained VDNNs with different training set sizes and compared the normalized mean absolute errors of the validation sets. In Figure 9.6b, we plot the normalized mean absolute errors of the validation sets as a function of training dataset size. From these plots, it is clear that the normalized mean absolute error converges as a function of the dataset size, and is well converged with a dataset size of 10^6 images. This value was used when training all reported models unless otherwise stated. We note that a single SCF step produces $n_x \times n_y \times n_z$ samples, where n denotes the number of real space grid points in a given direction. For the 32 atom graphene lattice, this number was 1.728×10^6. A single DFT calculation thus generates a large number of training examples and therefore very few DFT calculations are needed. We also see the slope of the line change at a dataset size of

$\approx 2.5 \times 10^5$ indicating a decrease in the rate of convergence. Based on this, one should use a minimum of 2.5×10^5 training examples to decrease the training time while maintaining accuracy. Again, this data can be easily extracted from DFT calculations.

How many calculations are needed to produce accurate kinetic energies? To answer this question, we trained VDNNs on the KS-DFT data and studied the accuracy of the models as a function of the number of atomic configurations. Specifically, we extracted a training dataset from 2, 4, 8, 16, 32, and 64 different training atomic configurations and calculated the mean absolute errors (MAEs), and root mean squared errors (RMSEs) of the kinetic energies for the testing set. It should be noted that a shift was applied to the predictions from VDNNs to obtain better results after integration. In a machine learning model, errors are never eliminated and become non-negligible after integrating on large numerical grids. A rigid shift on the *training* set rids the error accumulation on both the training and testing sets. In Figure 9.6c, we plot the MSE with their respective standard deviations. From the plot, we notice that error does not substantially decrease as a function of the number of atomic configurations. We, therefore, conclude that a model could be made from a training dataset with only 2 atomic configurations given that the MSE is less than chemical accuracy. Only 2 DFT calculations are needed to produce an accurate KED for pristine graphene lattices.

One of the major advantages of VDNNs is that they scale to arbitrary system size. After training a VDNN on the KS-DFT data, we ran calculations with the same kinetic energy cutoff (45 Ha) for 4, 8, 16, 32, and 64 atom unit cells. In Figure 9.6b, we show the absolute error of the predicted kinetic energy per electron and the inference time as a function of the number of atoms. From here, we see that the error remains constant as the number of atoms increases. In theory, VDNNs scale to an arbitrary system size with no increase in error per electron. The cost of inference scales linearly with the number of atoms (or number of grid points) in the system. The timings of the inference calculations were done with 16 nodes, each with 4 NVIDIA V100 GPUs.

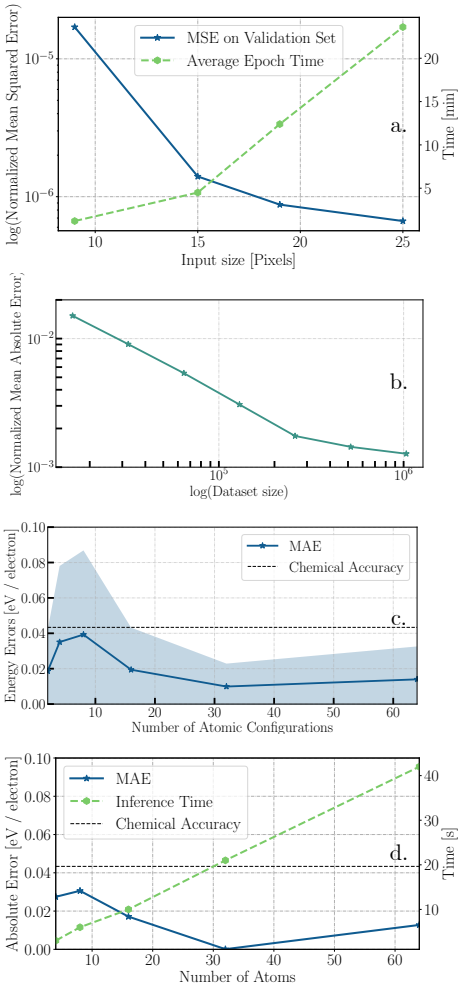


Figure 9.6: Convergence results for voxel deep neural networks. (a) The normalized mean squared error and the epoch time versus input size. (b) The normalized mean absolute error as a function of training dataset size. (c) The mean absolute error (line and points), and the root mean squared error (shaded region) as a function of the number of DFT training calculations. (d) The absolute error of the kinetic energy as a function of the number of atoms as well as the inference time versus the number of atoms. All kinetic energies shown here are from the Kohn-Sham non-interacting kinetic energy functional.

Chapter 10

Electronic Response Quantities of Solids and Deep Learning

In this Chapter, we build on Chapter 7 and Chapter 8, and extend our predictions to response quantities rather than limit them to ground state energies. Namely, we predict the polarization and infinite (or static) dielectric tensors of condensed matter systems. To do so, we introduce a deep learning framework that combines featureless deep learning with previously developed machine learning models used for quantum chemistry. We develop an atom-centered approach where 3D representations are constructed and mapped to deep convolutional neural networks that yield the contribution to a particular output. We use this methodology to compute various quantities of interest in response theory, including the Raman spectra. In doing so, we are able to compute quantities at a fraction of the cost compared to traditional calculations. This work also allows one to produce a surrogate model in a design pipeline to search for candidate structures with a desirable response property.

Kevin Ryczko
Department of Physics, University of Ottawa
Vector Institute for Artificial Intelligence

Olivier Malenfant-Thuot
Département de Physique, Université de Montréal
Regroupement Québécois sur les Matériaux de Pointe

Michel Côté
Département de Physique, Université de Montréal
Regroupement Québécois sur les Matériaux de Pointe

Isaac Tamblyn
Department of Physics, University of Ottawa
Vector Institute for Artificial Intelligence

Abstract

We introduce a deep neural network (DNN) framework called the **r**eal-space **a**tomic **d**ecomposition **net**work (RADNET), which is capable of making accurate polarization and static dielectric function predictions for solids. We use these predictions to calculate Born-effective charges, longitudinal optical transverse optical (LO-TO) splitting frequencies, and Raman tensors for two prototypical examples: GaAs and BN. We then compute the Raman spectra, and find excellent agreement with *ab initio* techniques. RADNET is as good or better than current methodologies. Lastly, we discuss how RADNET scales to larger systems, paving the way for predictions of response functions on meso-scale structures with *ab initio* accuracy.

10.1 Introduction

Advances in the application of machine learning in quantum chemistry and condensed matter physics have shown that one can accurately model the potential energy surfaces of molecules and materials [10, 147, 148, 64]. These machine learning models approach the accuracy of the electronic structure approach they were trained with, and demonstrate significant computational speed-ups compared to traditional electronic structure calculations. The next obvious application of machine learning is modeling how these potential energy surfaces change in response to a perturbation. In condensed matter physics, two commonly studied perturbations are atomic displacements and external electric fields. First-order responses of these quantities give forces and polarization. Second-order responses give phonons, electronic susceptibility, and effective charges. Third-order responses include Raman tensors, non-linear electronic susceptibility, and phonon-phonon interactions. A multitude of such quantities have yet to be reliably computed for solids with machine learning in a scalable manner.

For isolated molecules, various machine learning techniques have been introduced and applied to predict response quantities [149, 150, 151, 22]. In Ref. [149], a symmetry-adapted machine learning framework was introduced for the prediction of the dipole moment, polarization, and hyper-polarization of water oligomers. This methodology was also applied to the QM7b database [150], which contains coupled-cluster polarizabilities for various molecules. In Ref. [151], the operator quantum machine learning (OQML) framework was introduced based on kernel-based machine learning methods. This framework was used to compute forces, dipole moments, normal modes, and infrared spectra for different molecular datasets. In Ref. [22], the FieldSchNet architecture was introduced and utilized to calculate dipoles, polarizabilities, infrared spectra, Raman spectra, and nuclear magnetic resonance shifts for various molecules.

For solids, however, machine learning techniques have mostly been applied to responses for atomic displacements [152, 153, 154]. The only reports that focus on other response

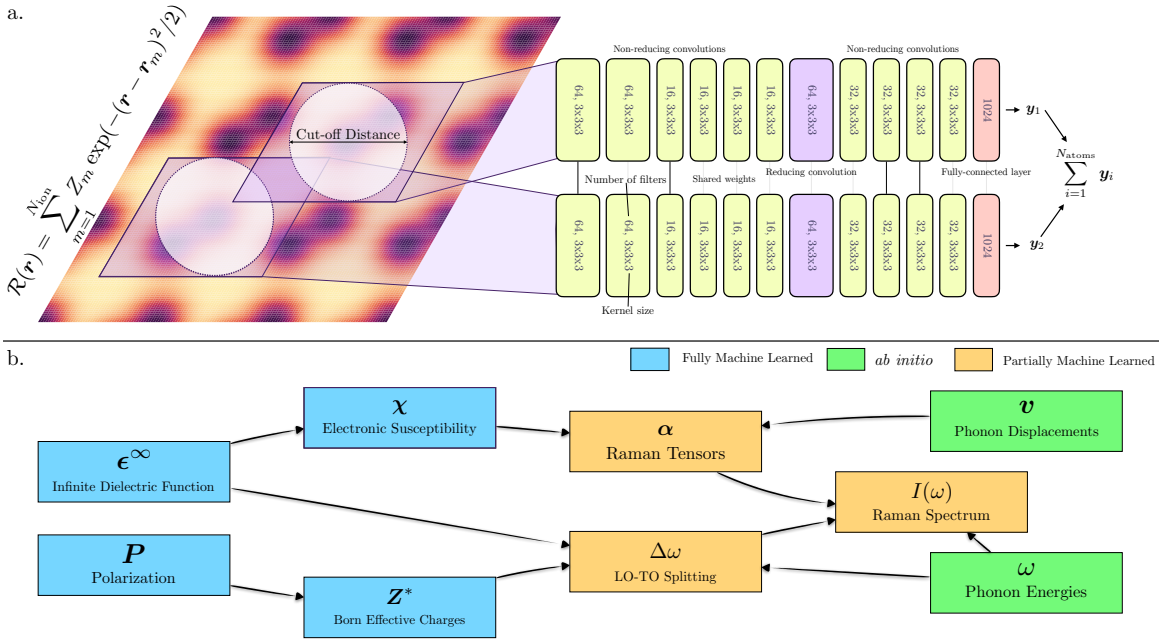


Figure 10.1: (a) RADNET. The system is first represented by a sum over Gaussian functions ($\mathcal{R}(\boldsymbol{r})$). 3D slices of $\mathcal{R}(\boldsymbol{r})$, centered on the atomic positions are passed through a deep neural network which outputs the atomic contribution to the desired quantity. (b) Quantities computed with either machine learning, *ab initio* techniques, or a hybrid of the two. Lines show the connections between the quantities.

quantities, to date, calculated dielectric tensors of liquid water and ice [149] or molecular crystals [155].

Here, we introduce a deep learning framework, called the **r**eal-space **a**tomic **d**ecomposition **net**work (RADNET), to predict the polarization and static dielectric tensor. We then use these predictions to calculate Born-effective charges, the longitudinal optical transverse optical (LO-TO) splitting, and Raman tensors. Although machine learning methods have been used to compute the polarization and electronic dielectric function in a condensed matter system [149], derivatives of these quantities have yet to be reported. In addition, previous reports focused on phonons for non-metallic systems which *did not* include LO-TO splitting. Therefore, our work allows for a machine-learned correction to LO phonon energies. LO-TO splitting occurs due to induced dipoles in the direction parallel to the applied electric field. In a solid, induced dipoles introduce long-range, dipole-dipole interactions and also

produce a macroscopic electric fields. Current work in the field is dedicated to constructing machine learning potentials that include long-range effects [156]. The outline of this report is as follows: In Section 10.2.1, we include all of the necessary electronic structure calculations needed to compute the various quantities of interest in this report. In Section 10.2.2, we cover the dataset generation details. In Section 10.2.3, we discuss machine learning methodology and hyper-parameter choices. In Section 10.3, we report and comment on our results and potential future work. Lastly, in Section 10.4, we summarize.

10.2 Methods

10.2.1 Electronic Structure Theory

We first discuss our *ab initio* computations, beginning with the polarization, $\boldsymbol{P}$. The polarization requires derivatives of Kohn-Sham orbitals with respect their wavevectors (which can be calculated via density functional perturbation theory (DFPT) [46] or Berry-phase theory [157]). In Berry-phase theory, as demonstrated in the modern theory of polarization [47], the total polarization per unit volume in atomic units is

$$\boldsymbol{P} = \frac{e}{(2\pi)^3}\mathrm{Im}\sum_{n=1}^{N_{\mathrm{el}}/2}\int_{\mathrm{BZ}} d\boldsymbol{k}\,\langle\psi_{n\boldsymbol{k}}|\nabla_{\boldsymbol{k}}|\psi_{n\boldsymbol{k}}\rangle + \frac{1}{\Omega_0}\sum_{m=1}^{N_{\mathrm{ion}}} Z_m \boldsymbol{r}_m, \tag{10.1}$$

where n is the band index, N_{el} is the number of electrons, $\psi_{n\boldsymbol{k}}$ is a Kohn-Sham orbital, Ω_0 is the unit cell volume, N_{ion} is the number of ions, and Z_m is the atomic number ion m. For multi-band systems, the electronic contribution is written in terms of overlap matrices between Kohn-Sham orbitals of neighbouring $\boldsymbol{k}$-points, as demonstrated in Ref. [158]. Once the polarization is known, the Born-effective charge, defined for a particular atom m

$$Z^*_{m,\beta\alpha} = \Omega_0 \frac{\partial P_\beta}{\partial \tau_{m,\alpha}(\boldsymbol{q}=0)} \tag{10.2}$$

can be computed with DFPT [159] or finite differences. In this work, we use finite differences to calculate derivatives of the the machine learning model predictions. In Equation 10.2, $\tau_{m,\alpha}$ is the perturbation to the atomic coordinate of atom m in the direction α.

Next, we discuss the computation of the electric susceptibility, $\boldsymbol{\chi}$, the static dielectric tensor, $\boldsymbol{\epsilon}^{\infty}$, and the Raman tensor, $\boldsymbol{\alpha}$. The electric susceptibility can be written as

$$\chi_{ij} = \frac{\partial^2 E}{\partial \mathcal{E}_i \partial \mathcal{E}_j} \tag{10.3}$$

where E is the total energy and $\boldsymbol{\mathcal{E}}$ is an electric field. The static electronic dielectric tensor can then be defined as

$$\epsilon_{ij}^{\infty} = \delta_{ij} - \frac{8\pi}{\Omega_0}\chi_{ij}, \tag{10.4}$$

where δ_{ij} is the Kronecker delta function. The Raman tensor adds an additional derivative to the energy with respect to an atomic displacement. For each phonon mode ℓ, the Raman tensor is defined as

$$\alpha_{ij}^{\ell} = \sum_{\alpha=1}^{3}\sum_{m=1}^{N_{\text{ion}}} \frac{\partial \chi_{ij}}{\partial \tau_{m,\alpha}} v_{\ell}(m,\alpha) \tag{10.5}$$

where and v_ℓ is a phonon eigenvector. Derivatives of the electronic susceptibility can also be computed using finite differences or DFPT. Using DFPT, the electronic susceptibility, as shown in [159], is computed via

$$\chi_{ij}(\boldsymbol{r}) = -\text{Im}\frac{\Omega_0}{(2\pi)^3}\sum_{n=1}^{N_{\text{el}}}\int_{\text{BZ}} d\boldsymbol{k}\ \langle u_{n\boldsymbol{k}}^{\mathcal{E}_i} | u_{n\boldsymbol{k}}^{k_j}\rangle, \tag{10.6}$$

where $u_{n\boldsymbol{k}}^{\mathcal{E}_i}$ is the first order correction to the ground state Kohn-Sham orbital $u_{n\boldsymbol{k}}$ with respect to an electric field $\boldsymbol{\mathcal{E}}$ applied in direction i, and $u_{n\boldsymbol{k}}^{k_j}$ is the periodic part of the Kohn-Sham orbital which has been differentiated with respect to its wavevector $\boldsymbol{k}$ in direction j.

With all the fundamental quantities now defined, we discuss the computation of LO-TO splitting and computation of the Raman spectrum. As shown in [159], the LO and and TO modes split due to an induced dipole in the direction of the applied electric field, and the difference between the squared phonon energies as $\boldsymbol{q} \to 0$ is

$$\Delta\omega(\boldsymbol{q}\to 0) = \frac{4\pi}{\Omega_0}\sum_{m=1}^{N_{\text{ion}}}\frac{1}{M_m}\frac{\sum_{\alpha\beta\gamma=1}^{3} q_\alpha Z_{m,\alpha\beta}^{*} Z_{m,\gamma\beta}^{*} q_\gamma}{\sum_{\alpha\beta=1}^{3} q_\alpha \epsilon_{\alpha\beta}^{\infty} q_\beta}, \tag{10.7}$$

where M_m is the mass of atom m. As shown in Ref. [160], the contribution of the phonon

mode ℓ, to the Raman spectrum is

$$I^{\ell}(\omega) = 2\pi C^{\ell}(\omega)[(10G_0^{\ell} + 4G_2^{\ell}) + (5G_1^{\ell} + 3G_2^{\ell})], \tag{10.8}$$

with

$$C^{\ell}(\omega) = \frac{(\omega_{\ell} - \omega_I)^4}{2\omega_{\ell}c^4}[n(\omega_{\ell}) + 1]\frac{\Gamma}{(\omega - \omega_{\ell})^2 + \Gamma^2}. \tag{10.9}$$

In Equation 10.8, the rotation invariants are [161]

$$G_0 = \frac{1}{3}(\alpha_{11}^2 + \alpha_{22}^2 + \alpha_{33}^2), \tag{10.10}$$

$$G_1 = \frac{1}{2}[(\alpha_{12} - \alpha_{21})^2 + (\alpha_{23} - \alpha_{32})^2 + (\alpha_{13} - \alpha_{31})^2], \tag{10.11}$$

$$G_2 = \frac{1}{2}[(\alpha_{12} + \alpha_{21})^2 + (\alpha_{23} + \alpha_{32})^2 + (\alpha_{13} + \alpha_{31})^2], \tag{10.12}$$

and in Equation 10.9, ω_{ℓ} is the frequency of the ℓth phonon, ω_I is the frequency of light, c is the speed of light, $n(\omega)$ is the Bose-Einstein distribution, and Γ is a broadening factor to simulate impurities that would be present experimentally. For the Bose-Einstein distribution, we used a temperature of 300 K. For ω_I, we used a value of 532 nm, which has been used in a previous report to study the Raman spectra of MoS_2 [162]. Additionally, we used $\Gamma = 10$ cm^{-1} for the broadening factor.

10.2.2 Data Generation

To demonstrate our approach, we focus on two example solids: GaAs and cubic BN. We defined 2 atom unit cells for both systems and computed the polarization and infinite dielectric tensor for different random configurations of the atomic positions. For GaAs, the lattice constant was 4.066 Å and for BN the lattice constant was 2.564 Å. The calculations were done using the PBE exchange-correlation functional, a $12 \times 12 \times 12$, unshifted $\boldsymbol{k}$-point grid, and an energy cutoff of 45 Ha. We used Abinit [163, 160, 164, 159] for all of our electronic structure calculations. To generate the random configurations, we used normal

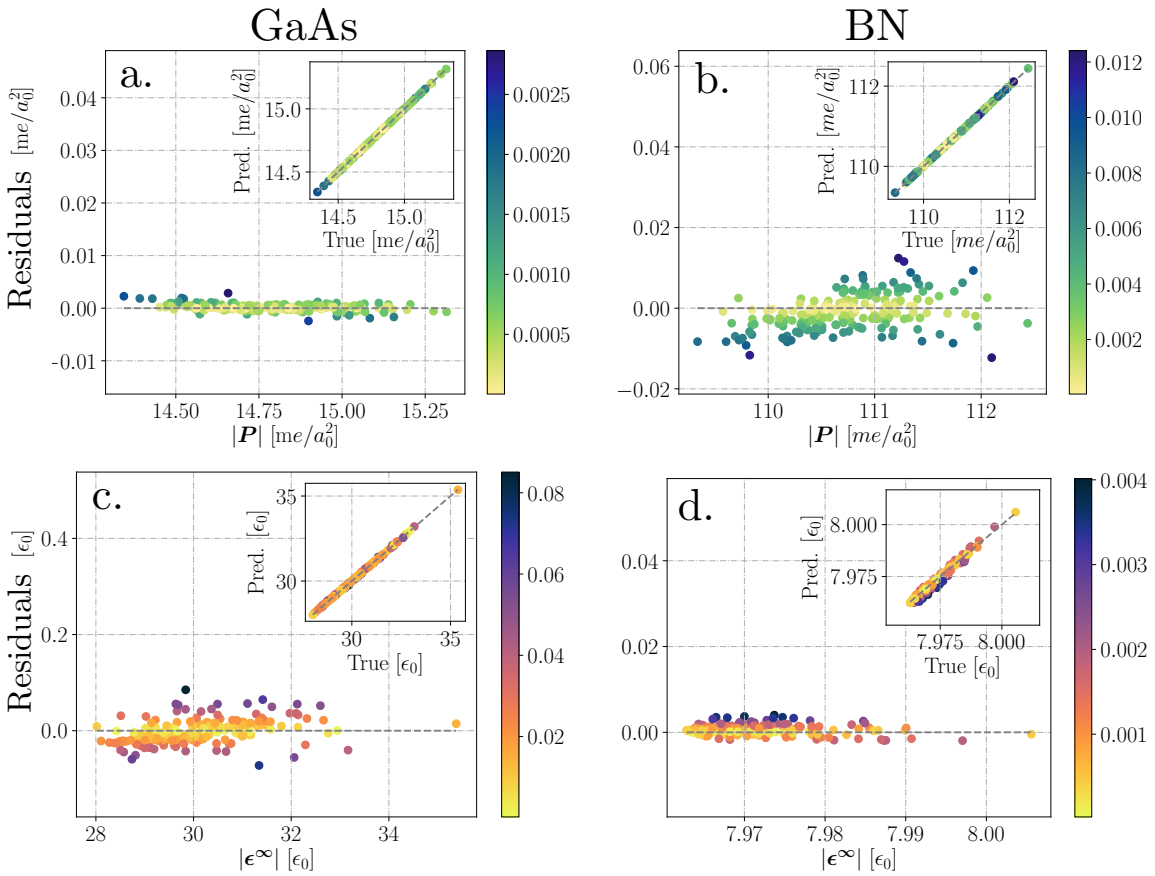


Figure 10.2: Residuals and true versus predicted (insert) for the absolute values of polarization for (a) GaAs and (b) BN and the static dielectric function for (c) GaAs and (d) BN. See Section 10.3 for a description of the computation of the absolute values.

distributions centered at the equilibrium positions with a standard deviation of 0.01 Å. The number of random configurations were 1200 for GaAs and 1000 for BN. In the additional 200 configurations for GaAs, the x-coordinates of the atoms were held fixed which allowed for an improved prediction of the infinite dielectric tensor. This is described in further detail in Section 10.3. Of the configurations, 20% were used as a validation set during training. The final tests were made with comparisons of the predicted quantities for the equilibrium structures unseen during training.

10.2.3 Deep Learning

As depicted in Figure 10.1, RADNET utilizes deep neural networks (DNNs) to calculate atomic contributions to the polarization and static dielectric tensor. Similar to Ref. [149], we write the vector or tensor quantity $\boldsymbol{Y}$ as

$$\boldsymbol{Y} = \sum_{m=1}^{N_{\text{ion}}} \boldsymbol{Y}_m. \tag{10.13}$$

The decomposition of the desired quantity into atomic contributions allows for an extensive DNN framework [103]. For the polarization, $\boldsymbol{Y}$ is a vector with 3 components. For the static dielectric tensor, $\boldsymbol{Y}$ is a vector with 6 components. These 6 components correspond to the non-equivalent elements of the tensor. Each component was independently normalized such that $Y_i \in [0, 1]$. For the input into the networks, we first construct a real-space representation by summing over Gaussian functions

$$\mathcal{R}(\boldsymbol{r}) = \sum_{m=1}^{N_{\text{ion}}} Z_m \exp(-(\boldsymbol{r} - \boldsymbol{r}_m)^2/2). \tag{10.14}$$

This representation has had success in the past for machine-learned potential energy surfaces [12, 17, 142, 103]. The 3-dimensional grid on which Equation 10.14 is evaluated must be chosen such that the evaluation time of $\mathcal{R}$ is minimal but the information is not hampered by a poor spatial resolution. In our case, we halved the grid sizes used in the DFT calculations. For GaAs the grid size used was $24 \times 24 \times 24$, and for BN the grid size used was $15 \times 15 \times 15$. We then locate the voxels that correspond to the atomic positions and take 3-dimensional slices of the function $\mathcal{R}(\boldsymbol{r})$, centered on atomic positions. The dimensions of the slice are defined by a specified cut-off radius, which is used to define a spherical field of view, as shown in Figure 10.1a. We experimented with the cut-off radius when predicting $\boldsymbol{\epsilon}^{\infty}$ of GaAs and found that a value of 1.852 Å (3.5 Bohr) yielded the lowest error on the validation set. We used this cut-off radius for all subsequent calculations. The spherical field of view is achieved by applying a spherical cut-off function, where we considered both a hard cut-off and a smooth one (the complimentary error function). We compared the performance of a model predicting $\boldsymbol{\epsilon}^{\infty}$ for GaAs with each cut-off function. We found that both functions

performed similarly and chose to use the smooth one. After applying the spherical cut-off function, the slices were fed into a DNN that outputs the contribution to the desired quantity. For the DNN architecture, we used the same DNN as used in Ref. [165]. It is comprised of a set of reducing and non-reducing 3-dimensional convolutional layers, followed by a fully connected layer that outputs the atomic contribution to the quantity of interest. The final output is found by summing over the atomic contributions. When training, we used a batch size of 32, a learning rate of 10^{-5}, and trained for 10^4 epochs. The models that had the lowest mean squared errors on the validation sets were used for inference. All models were implemented in PyTorch [166]. Datasets and codes can be found here [145].

10.3 Results

We now discuss results of the machine learning models for the computation of the polarization $\boldsymbol{P}$ and static dielectric tensor $\boldsymbol{\epsilon}^{\infty}$. In Figure 10.2, we show the true versus predicted values for $|\boldsymbol{P}|$ and $|\boldsymbol{\epsilon}^{\infty}|$ for the different systems. For $\boldsymbol{P}$, we compute the absolute value via

$$|\boldsymbol{P}| = \sqrt{P_1^2 + P_2^2 + P_3^2} \tag{10.15}$$

and for $\boldsymbol{\epsilon}^{\infty}$ we use

$$|\boldsymbol{\epsilon}^{\infty}| = \sqrt{(\epsilon_{11}^{\infty})^2 + (\epsilon_{12}^{\infty})^2 + (\epsilon_{13}^{\infty})^2 + (\epsilon_{22}^{\infty})^2 + (\epsilon_{23}^{\infty})^2 + (\epsilon_{33}^{\infty})^2}. \tag{10.16}$$

In all 4 plots, we see excellent agreement between the true and predicted values. RADNET can accurately make predictions despite the changes in the ranges of values of $\boldsymbol{P}$ and $\boldsymbol{\epsilon}^{\infty}$. In the case of GaAs, the range of $|\boldsymbol{\epsilon}^{\infty}|$ is greater than $|\boldsymbol{P}|$ ($\approx$ 8x), whereas for BN we observe the opposite behavior ($\approx$ 80x). For BN, $\boldsymbol{\epsilon}^{\infty}$ varies minimally with the random perturbations of the atomic positions. When comparing the errors in Table 10.1 for $\boldsymbol{P}$ for GaAs and BN, we find that errors are of similar magnitude. Errors in Table 10.1 for $\boldsymbol{\epsilon}^{\infty}$ for GaAs and BN show that BN errors are an order of magnitude smaller than GaAs. However, if we consider the error relative to the range of values shown in Figure 10.2, we find that the relative error for GaAs is two orders of magnitude smaller. The quantities are driven both by the model

System	Description	Quantity	MAE	RMSE
GaAs	Val. Set	$\boldsymbol{P}$ $[e/a_0^2]$	4.25×10^{-4}	6.78×10^{-4}
GaAs	Val. Set	$\boldsymbol{\epsilon}^{\infty}$ $[\epsilon_0]$	$\mathbf{1.25 \times 10^{-2}}$	$\mathbf{1.77 \times 10^{-2}}$
GaAs	SchNet	$\boldsymbol{\epsilon}^{\infty}$ $[\epsilon_0]$	0.38	0.51
GaAs	SchNet-α	$\boldsymbol{\epsilon}^{\infty}$ $[\epsilon_0]$	7.61	8.93
GaAs	SA-GPR	$\boldsymbol{\epsilon}^{\infty}$ $[\epsilon_0]$	1.03	1.23
GaAs	Equil.	$\boldsymbol{P}$ $[e/a_0^2]$	1.19×10^{-6}	1.32×10^{-6}
GaAs	Equil.	$\boldsymbol{\epsilon}^{\infty}$ $[\epsilon_0]$	0.65	0.72
GaAs	Equil.	$\boldsymbol{Z}^*$ $[e]$	1.04×10^{-2}	1.43×10^{-2}
GaAs	Equil.	$\boldsymbol{\alpha}$ $[\epsilon_0/a_0^3]$	0.30	0.46
GaAs	Equil.	LO-TO $[\mathrm{cm}^{-1}]$	0.33	-
BN	Val. Set	$\boldsymbol{P}$ $[e/a_0^2]$	3.85×10^{-3}	5.18×10^{-3}
BN	Val. Set	$\boldsymbol{\epsilon}^{\infty}$ $[\epsilon_0]$	5.19×10^{-3}	7.32×10^{-3}
BN	SchNet	$\boldsymbol{\epsilon}^{\infty}$ $[\epsilon_0]$	1.11×10^{-2}	1.53×10^{-2}
BN	SchNet-α	$\boldsymbol{\epsilon}^{\infty}$ $[\epsilon_0]$	1.85	2.17
BN	SA-GPR	$\boldsymbol{\epsilon}^{\infty}$ $[\epsilon_0]$	$\mathbf{1.38 \times 10^{-3}}$	$\mathbf{2.18 \times 10^{-3}}$
BN	Equil.	$\boldsymbol{P}$ $[e/a_0^2]$	1.68×10^{-6}	1.91×10^{-6}
BN	Equil.	$\boldsymbol{\epsilon}^{\infty}$ $[\epsilon_0]$	2.04×10^{-4}	2.18×10^{-4}
BN	Equil.	$\boldsymbol{Z}^*$ $[e]$	5.65×10^{-3}	6.52×10^{-3}
BN	Equil.	$\boldsymbol{\alpha}$ $[\epsilon_0/a_0^3]$	2.80×10^{-4}	4.20×10^{-4}
BN	Equil.	LO-TO $[\mathrm{cm}^{-1}]$	2.14	-

Table 10.1: Mean absolute errors (MAEs) and root mean absolute errors (RMSEs) for GaAs and BN for different quantities (units in square brackets) in different settings. The description "Val. Set" means validation set and "Equil." means equilibrium structure. SchNet, SchNet-α, and symmetry-adapted Gaussian process regression (SA-GPR) values are for the validation set. SchNet-α signifies that the polarizability(dielectric) output module was used. Best of the model comparisons are indicated in bold.

error as well as the range of values being predicted.

To compare our methodology with atom-centered approaches, we consider SchNet [19] and symmetry adapted Gaussian process regression (SA-GPR) [149]. Errors associated with validation sets of these models are reported in Table 10.1. In the case of SchNet, we used an output module meant for energy predictions, which we refer to as SchNet, and an output module specifically designed for molecular polarizabilities (dielectric function), which we refer to as SchNet-α. In all cases, we used the default parameters included with these

packages and radial cut-offs of 5 Å. We found that SchNet outperforms SchNet-α for both GaAs and BN. In the case of GaAs, SchNet is the best performing alternative to RADNET. In the case of BN, SA-GPR outperforms both SchNet and RADNET. SchNet is an order of magnitude worse and RADNET is on-par with SA-GPR. We find the performance of RADNET is better or on-par with pre-existing methods. However, an extensive hyperparameter search was not performed for the pre-existing methods.

When predicting $\boldsymbol{P}$ and $\boldsymbol{\epsilon}^{\infty}$ for the equilibrium structures, we find that the errors are less than the validation sets in all cases except when predicting $\boldsymbol{\epsilon}^{\infty}$ for GaAs. For both GaAs and BN, $\boldsymbol{\epsilon}^{\infty}$ for the equilibrium structure is isotropic. Off-diagonal elements are only non-zero when symmetry is broken. This is true for all training and validation samples in the BN dataset. As discussed in Section 10.2, for the GaAs dataset, an additional 200 calculations were included where the x-coordinates of the atoms were held fixed. This resulted in more configurations with zero off-diagonal elements. This effect is also true for BN, but the magnitude of the off-diagonal elements is less than those of GaAs. Careful consideration of the desired quantities must be done when constructing an optimal training set.

We now discuss the Born-effective charges, $\boldsymbol{Z}^*$. As mentioned in Section 10.2, we compute $\boldsymbol{Z}^*$ with finite differences with $\Delta = 0.026$ Å(0.05 Bohr). We also enforce charge neutrality of $\boldsymbol{Z}^*$ by following the sum rules outlined in Ref. [159]. Looking to Table 10.1, we note that for both GaAs and BN, the errors for $\boldsymbol{Z}^*$ are 3-4 orders of magnitude larger than their respective $\boldsymbol{P}$ errors. There are two contributions to this error. The first is the difference in ranges for these quantities. The range-driven error causes the magnitude of $\boldsymbol{Z}^*$ to be 2-3 orders of magnitude larger than $\boldsymbol{P}$ which accounts for the majority of the error differences. The remainder of the error can be attributed to errors associated with the machine learning model. In some cases, over-fitting can be reduced with the use of regularization techniques as was shown in a previous report for phonon calculations [167]. However, in our case, we found that the use of $L2$ regularization did not improve the derivative predictions and therefore conclude that over-fitting is not a factor. The remainder of the error could be improved by an improved DNN architecture. However, the current accuracy of the

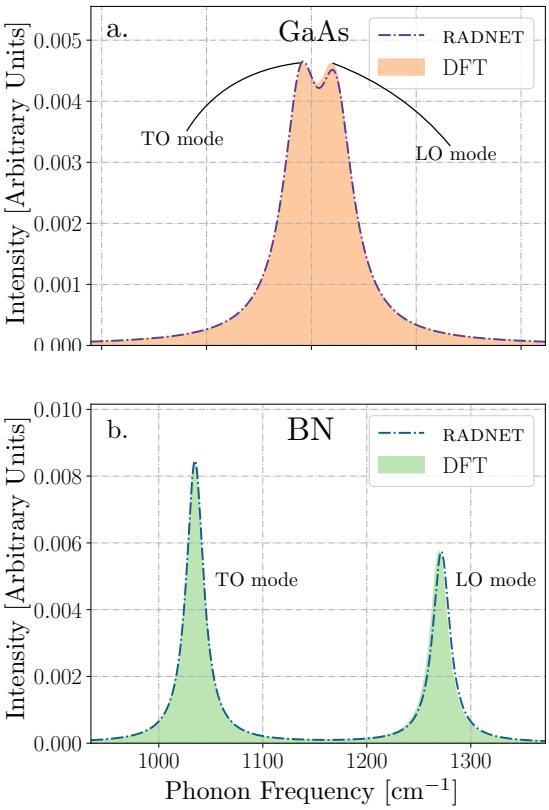


Figure 10.3: Raman spectra calculated via DFT (shaded regions) and via ML (dashed lines) for (a) GaAs and (b) BN.

predictions is satisfactory and its use in subsequent quantities yielded good agreement with conventional approaches.

We now discuss LO-TO splitting. In Equation 10.7, the TO phonon energy is needed to calculate the splitting, which yields the LO energy. We used the DFT phonon energies from the equilibrium structures along with the machine-learned quantities to calculate the splitting. We note that one can also use various machine learning techniques to calculate the phonon energies if one has a differentiable model which explicitly depends on atomic coordinates. We find that errors of the LO modes for both GaAs and BN are less than 0.2%. In a past report [168], phonon energies calculated via DFT with different hybrid functionals were found to have maximum errors of $\approx 5\%$ when compared to experiment. The predictions we report are well within this range. Despite having a large error in the prediction of ϵ^{∞}

for the equilibrium structure of GaAs, we achieve excellent agreement when comparing the LO-TO splitting frequencies. This is due to the choice of the vector $\boldsymbol{q}$ and the form of Equation 10.7. The choice of $\boldsymbol{q}$-vector when calculating the LO-TO splitting is arbitrary, but reasonable choices are the Cartesian unit vectors. This particular choice eliminates the error from off-diagonal elements of $\boldsymbol{\epsilon}^{\infty}$ due to the denominator in Equation 10.7. With $\boldsymbol{q} = \hat{x}$ Bohr^{-1}, the first diagonal term is selected. With $\boldsymbol{q} = \hat{y}$ Bohr^{-1} the second diagonal term, and with $\boldsymbol{q} = \hat{z}$ Bohr^{-1} the third diagonal term. If one is only interested in computing the Raman spectra for these choices of $\boldsymbol{q}$-vectors then the off-diagonal terms can be omitted entirely from the datasets. The value of the LO-TO splitting frequency, as reported in Table 10.1, is averaged over the 3 unit directions.

We now discuss computation of the Raman spectra. Using finite differences, we compute the derivatives with respect to atomic displacement, thereby yielding the Raman tensors, $\boldsymbol{\alpha}$ shown in Equation 10.5. We used the DFT phonon eigenvectors from the equilibrium structures along with the machine-learned quantities to calculate $\boldsymbol{\alpha}$. As noted previously, if one has a fully differentiable machine learning model that explicitly depends on atomic coordinates, it is possible to calculate the phonon eigenvectors using automatic differentiation. In Table 10.1, we report errors of $\boldsymbol{\alpha}$. For both GaAs and BN, we find that these errors are on-par with their respective $\boldsymbol{\epsilon}^{\infty}$ counterparts. Using the phonon energies calculated via DFT, along with the LO-TO splitting and $\boldsymbol{\alpha}$ values calculated via machine learning, we plot the Raman spectra for GaAs and BN in Figure 10.3. Agreement of the LO-TO splitting is observed once again for both GaAs and BN when comparing the LO modes. In addition, the peak heights are also in good agreement due to the accurate predictions of $\boldsymbol{\alpha}$. This is promising for future studies involving larger-scale systems. In addition to the excellent agreement, the computational speed-up from our inference script in comparison to a DFT calculation of $\boldsymbol{\epsilon}^{\infty}$ was $\approx$250 times faster. Inference timing includes checkpoint loading, input generation, model inference, and derivative calculations (which were all done on 32 CPU cores). This accuracy and computational speed-up show promise to study the Raman response of large-scale systems. The relative height changes of peaks with respect to change in defect concentration have been done experimentally [162] for MoS_2, and a future application of RADNET will be

studying the effects of defects in the Raman response of solids. To do so, two limitations of RADNET must be addressed. The first limitation is the used of finite differences to evaluate derivatives. In order to perform inference on larger-scale systems, automatic differentation must be integrated. The second limitation is the implementation of the input function generation $\mathcal{R}(\boldsymbol{r})$, which is currently evaluated on the entire grid. Due to the locality of Gaussian functions, a radial cut-off can also be applied when evaluating the Gaussian on a numerical grid to improve the evaluation time. Therefore, an efficient, parallel implementation for the input function generation, similar to what was used in Ref. [103], must also be implemented.

10.4 Conclusion

We introduce a deep learning approach, called the **r**eal-space **a**tomic **d**ecomposition **net**work, or RADNET, for predicting the polarization and the static dielectric tensor in solids. We find excellent agreement when comparing predictions to true values and the relative error for both GaAs and BN are of similar magnitude. We also compare our results to other approaches when considering the dielectric function, and find that RADNET has the best result for GaAs, and is on-par with symmetry-adapted Gaussian process regression [149] for BN. After performing inference to obtain the polarization and infinite dielectric tensor, we then used these predictions to calculate Born-effective charges, the LO-TO splitting, and Raman tensors. We find good agreement when comparing the machine-learned quantities to DFT, and used the LO-TO splitting frequencies and Raman tensors along with DFT phonon energies and eigenvectors to compute the Raman spectra. The LO-TO splitting agreement is again confirmed when comparing the Raman spectra computed with the machine-learned quantities and DFT. Shifts of the LO modes as well as the peak heights are in exceptional agreement. Future work will include studying condensed matter systems with defects to examine the effects on the Raman spectra of realistic defect concentrations.

10.5 Acknowledgements

The authors acknowledge fruitful discussions with Arnab Majumdar. The authors also acknowledge Compute Canada, the Artificial Intelligence for Design (AI4D) program at the National Research Council, and the Vector Institute for Artificial Intelligence for computational resources. The authors also acknowledges the National Sciences and Engineering Council of Canada. KR acknowledges the Vector Institute for Artificial Intelligence for funding.

Chapter 11

Toward Acceleration of Quantum Monte Carlo with Deep Learning

In this chapter, we build on Chapter 9 by applying voxel deep neural networks (VDNNs) to diffusion Monte Carlo (DMC) calculations. DMC is exact, and is known to achieve highly accurate internal energies. This is due to the explicit inclusion of electron-electron interactions within the calculations. However, DMC calculations are computationally demanding and therefore large datasets computed with DMC are absent in the literature. The use of VDNNs eliminates this problem, as shown in Chapter 9. We therefore apply VDNNs to DMC calculations in an effort to accelerate the computation and improve the accuracy for properties of atomistic systems. Having a fast and highly accurate surrogate model would be invaluable for the design on next-generation materials. This work is a step in the right direction toward this realization.

Kevin Ryczko
Department of Physics, University of Ottawa, Ottawa, Ontario, Canada
Vector Institute for Artificial Intelligence, Toronto, Ontario, Canada

Jaron T. Krogel
Materials Science and Technology Division, Oak Ridge National Laboratory, Oak Ridge, Tennessee, United States

Isaac Tamblyn
National Research Council of Canada, Ottawa, Ontario, Canada
Department of Physics, University of Ottawa, Ottawa, Ontario, Canada
Vector Institute for Artificial Intelligence, Toronto, Ontario, Canada

Abstract

We use voxel deep neural networks (VDNNs) to map electron densities computed via density functional theory (DFT) to diffusion Monte Carlo (DMC) energy densities. We use graphene lattices as prototypical demonstrations. We show that a few QMC calculations are required to obtain predictions within chemical accuracy for pristine graphene lattices with random atomic displacements. Additionally, we use VDNNs to compute the energy barrier associated with a Stone-Wales defect and compare it with DFT and QMC calculations.

11.1 Introduction

Determining total internal energies and how they change with respect to some perturbation are central objectives in both quantum chemistry and condensed matter physics. Internal energies contribute to the stability of an atomistic system, and derivatives with respect to different variables yield a multitude of desired quantities. Recently, machine learning has been used in a variety of ways to calculate total electronic energies for toy models [14, 142, 143, 134], molecular systems [11, 13, 169, 19, 17, 30], and solid state systems [170, 142, 103, 134, 165]. Thus far, a majority of these works have focused on learning Hartree-Fock (HF) or density functional theory (DFT) computed properties. These models can make accurate and rapid predictions, but their utility is limited to the accuracy of the electronic structure method they were trained with. When studying atomistic systems, one must properly describe electron correlation to achieve high accuracy. Such a treatment leads to computationally expensive electronic structure methods. For molecules, common techniques include Møller-Plesset perturbation theory (MP2) or coupled-cluster singles, doubles and selected triples (CCSD(T)). For solid-state systems, the gold standard is quantum Monte Carlo (QMC).

Several studies have utilized machine learning to predict quantities computed with MP2 or CCSD(T) [150, 171, 172, 173, 174]. In Ref. [171] a novel representation, called the density tensor representation, was introduced and was used to predict accurate energies and dipoles of small molecules at the MP2 level. In Ref. [174], the density Δ-DFT approach was introduced and utilized for small molecules. This methodology used the DFT electronic density as input to a machine learning model to predict a difference in energy between CCSD(T) and DFT, allowing for a molecular dynamics simulation with quantum chemical accuracy. Training on the difference between DFT and CCSD(T), rather than the absolute value, allowed for an improved model accuracy when performing inference (predictions).

However, machine learning studies for solid-state systems with accurate electronic structure calculations are currently absent in the literature. This is due to the limitations of typical

machine learning implementations, the computational cost of the electronic structure calculations, and the lack of abundant QMC data for solid-state systems. Most current machine learning approaches require large quantities of training data, making it infeasible to train on properties calculated with QMC. However, recent machine learning methodologies have eliminated this problem by focusing on scalar functions, rather than scalar quantities [30, 134, 165]. Namely, it was shown that electron kinetic energies could be predicted within chemical accuracy from only 2 DFT calculations using a technique called voxel deep neural networks (VDNNs) [165].

In this letter, we utilize VDNNs to compute QMC energy densities for solid-state systems by mapping an electron density computed via DFT to kinetic, electron-electron, and electron-ion energy densities computed with diffusion Monte Carlo (DMC). To date, this is the first report that calculates QMC energies for a solid-state system with supervised machine learning.

11.2 Methods

The solid-state system we focus on is graphene with and without Stone-Wales defects. In Kohn-Sham DFT [41] the electron density is written as

$$\rho(\boldsymbol{r}) = 2\sum_{n}^{\text{occ}}\sum_{\boldsymbol{k}} w_{\boldsymbol{k}}\psi^{*}_{n,\boldsymbol{k}}(\boldsymbol{r})\psi_{n,\boldsymbol{k}}(\boldsymbol{r}). \tag{11.1}$$

In Equation 11.1, n is the band index, $\boldsymbol{k}$ is the k-point, $w_{\boldsymbol{k}}$ is the weighting associated with the k-point and ψ is a Kohn-Sham orbital. We focus on 50 atom graphene sheets in a non-orthorhombic supercell with supercell vectors $a_1 = (12.310, 0, 0)$, $a_2 = (-6.155, 10.661, 0)$, and $a_3 = (0, 0, 6.000)$ Å. The QMC electronic contribution to the energy density, as derived in Ref. [175] is

$$\mathcal{E}(\boldsymbol{r}) = \mathcal{T}(\boldsymbol{r}) + \mathcal{V}_{\text{ee}}(\boldsymbol{r}) + \mathcal{V}_{\text{ei}}(\boldsymbol{r}), \tag{11.2}$$

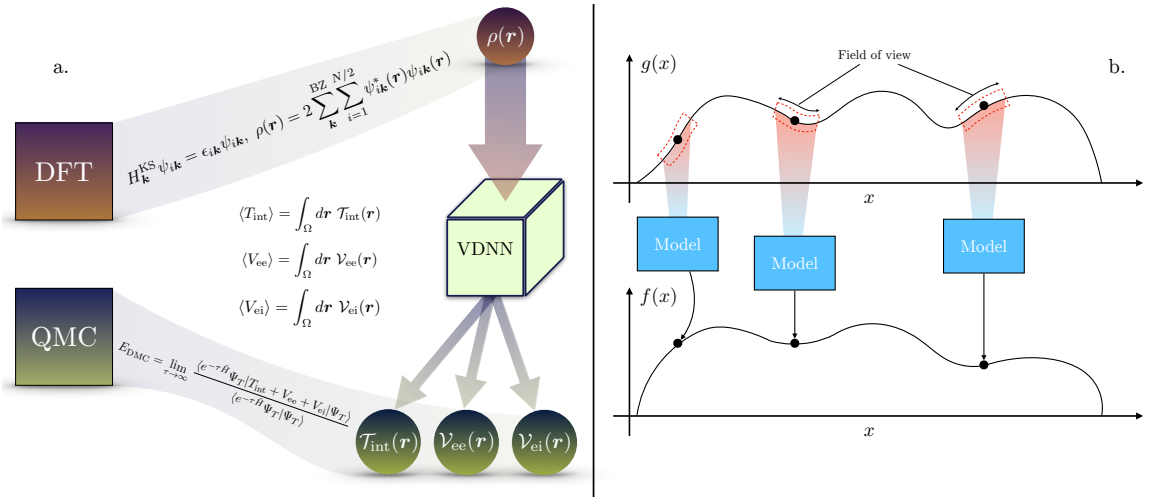


Figure 11.1: Visual representation of the methodology used here. In (a) we show how a voxel deep neural network is used to map an electron density computed via density functional theory to energy densities computed via diffusion Monte Carlo. These include the interacting kinetic energy density ($\mathcal{T}_{\text{int}}$), the electron-electron energy density ($\mathcal{V}_{\text{ee}}$), and the electron-ion energy density ($\mathcal{V}_{\text{ei}}$). The Hamiltonian $\hat{H} = T_{\text{int}} + V_{\text{ee}} + V_{\text{ei}}$, Ψ_T is the guiding many body wavefunction, and τ is imaginary time. In (b) we show a 1D example of a voxel deep neural network. A model is used to map the function $g(x)$ to another function $f(x)$. For every value x, there is a fixed field of view that is used to construct inputs for the machine learning model.

where the many-body kinetic energy density is

$$\mathcal{T}(\boldsymbol{r}) = -\frac{1}{2}\sum_{i=1}^{N}\int d\boldsymbol{R}\,\delta(\boldsymbol{r}-\boldsymbol{r}_i)\Psi^*(\boldsymbol{R})\nabla_i^2\Psi(\boldsymbol{R}), \tag{11.3}$$

and the many-body potential energy densities are

$$\mathcal{V}_{\text{ab}}(\boldsymbol{r}) = \frac{1}{2}\sum_{i=1}^{N_a}\sum_{j=1}^{N_b}\int d\boldsymbol{R}\,\delta(\boldsymbol{r}-\boldsymbol{r}_i)|\Psi(\boldsymbol{R})|^2 v_{\text{ab}}(\boldsymbol{r}_i,\boldsymbol{r}_j). \tag{11.4}$$

In Equations 11.3 and 11.4, $\boldsymbol{R} \equiv \{\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_N\}$ is the full many-body space, v_{ab} is the interaction potential between two species, and Ψ is the many-body wavefunction. For $\mathcal{V}_{\text{ee}}$, the interaction potential v_{ee} is the Coulomb potential. For the potential energy density term $\mathcal{V}_{\text{ei}}$, the interaction potential is a norm-conserving pseudopotential of the Burkatzki-Filippi-Dolg form [176]. The many-body wavefunction is constructed from a Slater determinant of Kohn-Sham orbitals and an optimized Jastrow factor. The QMC energy densities are

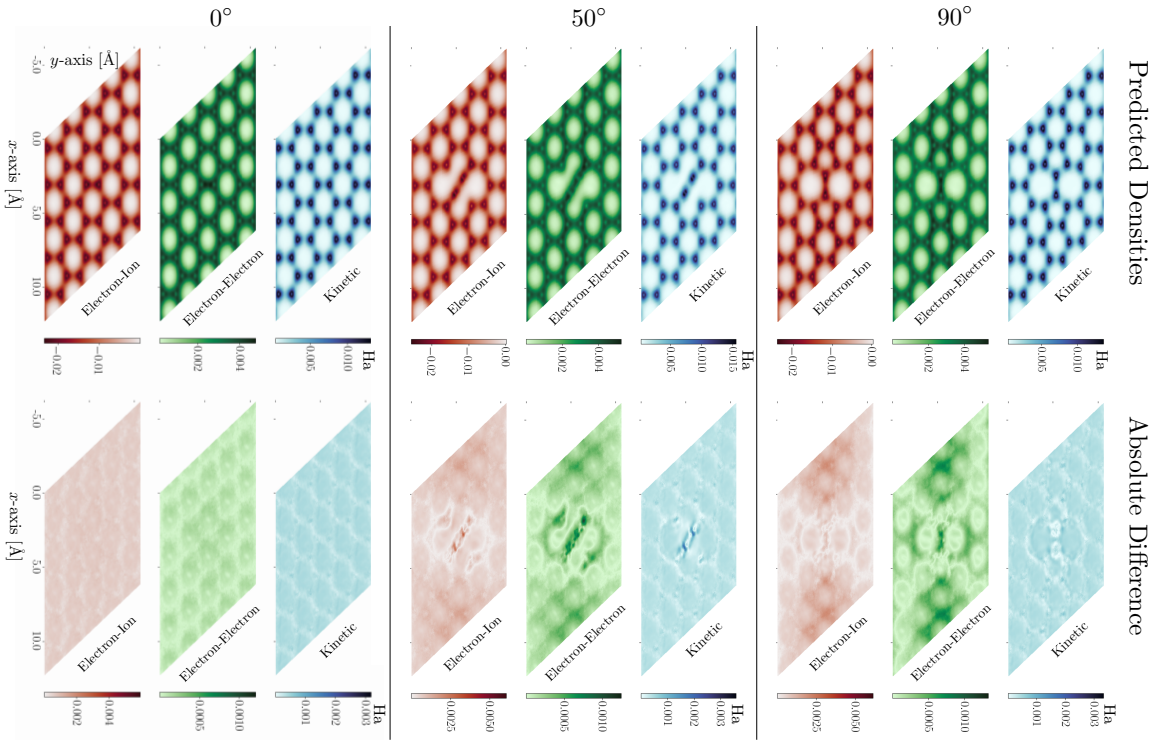


Figure 11.2: Predicted energy densities, and absolute differences between true and predicted energy densities with various angles. The angle corresponds to the rotation of the middle two atoms within the supercell. The angle $0°$ corresponds to a pristine graphene lattice and the angle $90°$ corresponds to a graphene lattice with a Stone-Wales defect.

generated in 3 stages. The first stage is a DFT calculation to obtain the Kohn-Sham orbitals needed for the Slater determinant. To perform the DFT calculation, we used Quantum Espresso [177] included with QMCPACK [178] with an energy cut-off of 150 Ha and a $4 \times 4 \times 1$ k-point grid. The second stage is the Jastrow factor optimization. We used a one-body Jastrow factor for the nuclei and a two-body Jastrow factor for the electrons. The final stage is evaluating the energy which was done in 3 steps. This included a variational Monte Carlo (VMC) calculation followed by two DMC calculations, each with different time steps. We refer the reader to the SI for more information on the Jastrow optimization and other QMC parameters Section 11.A. All three stages were performed at each $\boldsymbol{k}$-point, and the final energy densities were obtained by averaging over all $\boldsymbol{k}$-points. All calculations were performed using QMCPACK [178], Nexus [179], and a custom Python library [145].

When generating a dataset, we used $\approx 5\times10^5$, $19\times19\times19$ slices of the DFT electron densities that were collected from the training configurations such that a uniform distribution of the function $\mathcal{F} = \sqrt{\mathcal{T}^2 + \mathcal{V}_{\text{ee}}^2 + \mathcal{V}_{\text{ei}}^2}$ was produced. These parameter choices were found to be optimal in a previous study on the prediction of DFT kinetic energy densities [165]. Of these electron density slices, 99% were used for training and 1% were used for validation. When training the VDNNs, we map the electron density slices to values of $\mathcal{T}$, $\mathcal{V}_{\text{ee}}$, and $\mathcal{V}_{\text{ei}}$, as shown in Figure 11.1a. We used the convolutional neural network (CNN) from Ref. [165], a batch size of 512, a learning rate of 10^{-5}, the Adam optimizer [60], and layer-wise adaptive rate scaling with clipping [56]. We use ensemble models, where 5 independent networks were trained simultaneously and total energies were averaged over the models. We trained the networks for 250 epochs, and used the model that had the minimal mean squared error on the validation set. We found that the models began to overfit beyond ≈ 250 epochs. This overfitting can be attributed to the stochastic nature of the underlying training set. Beyond a certain point in training, the model begins to fit to statistical noise. We stopped training prior to this and used the model with the lowest mean squared error on the validation set. Training and inference was done across multiple nodes, each with 4 NVIDIA V100 GPUs. Our codes can be found here [145].

11.3 Results

The first task is the prediction of QMC total energy for pristine graphene lattices with random atomic perturbations. When building this dataset, we generated 8 atomic configurations where random perturbations of the atomic positions were applied to the equilibrium structure. To generate the random displacements, uniform random numbers were drawn with the range $[-0.1, 0.1]$ and were added to the pristine coordinates. 4 of these calculations were used to generate training/validation data and the remaining 4 were used for testing. Next, we trained 4 independent ensemble models, where each model was allowed to see 1, 2, 3 or 4 training configurations. We then computed the total energies of the testing set by integrating the predicted energy densities from the ensemble models. In Figure 11.2, true

and predicted densities are shown for a pristine lattice. Similar to Ref. [165], a rigid shift was applied to the predictions based on the residuals of the training set. When comparing the residuals of the energy densities, they are not symmetric about 0, and therefore accumulate after integration has been done on large grids. The shift can be calculated by computing the average error within an energy window. The number of energy windows depends on the structure of the residuals, and therefore differs for each model within the ensemble. We used a grid search to compute the optimal number of energy windows, and found the number of energy windows to be in the range $[8, 32]$. The ensemble model with all 4 training configurations had the lowest mean absolute error (MAE) and root mean absolute error (RMSE), which was 29.9 meV / electron and 33.7 meV / electron, respectively. Comparing this model to the ensemble model trained from just 1 configuration, we find a $\approx$ 3x improvement to the MAE. However, comparing this model to the ensemble model trained from 3 configurations, we find only a 2% improvement, with MAE and RMSE of 30.6 meV / electron and 34.5 meV / electron respectively. These results can be seen in Section 11.A. We therefore conclude that for a pristine graphene lattice with random atomic displacements, chemical accuracy can be achieved from only *3 QMC calculations*. However, this performance is limited to pristine graphene lattices and the model would require further training to handle environments with differing electron densities.

In the second task we calculate the energy barrier curve associated with the introduction of a Stone-Wales defect with a VDNN, which was previously investigated with DFT in Ref. [180]. As shown in Figure 11.3, by rotating two atoms in the graphene lattice, one transitions from a pristine lattice ($\theta = 0°$) to a lattice with a Stone-Wales defect ($\theta = 90°$). For the training set, we used 2 configurations, each with a single Stone-Wales defect and random atomic perturbations. The testing/validation dataset consisted of 10 configurations, where two atoms were rotated such that the nth configuration had an angle $\theta = n\pi/18,\ n \in [1, 9]$. For each configuration, a DFT structural relaxation was performed while holding the two rotated atoms fixed. As done in Ref. [180], the bond length between the two atoms rotated was also reduced. When calculating the rigid shifts based on the residuals, we found the optimal number of energy bins for each model based on the MAEs of the total electronic

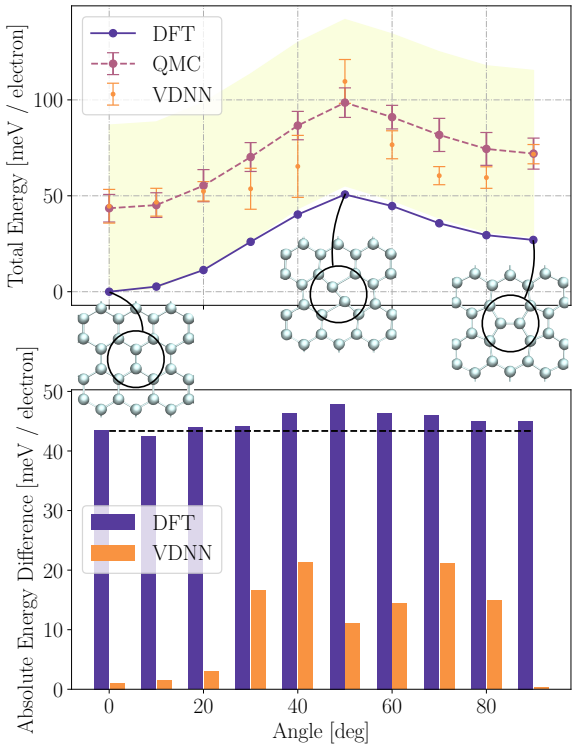


Figure 11.3: Top: Total energies computed with DFT, QMC, and VDNNs as a function of angle when rotating 2 atoms in the graphene lattice. Bottom: Absolute energy difference between QMC and DFT and QMC and VDNNs. A VDNN has a large reduction in error in comparison with DFT. With an angle of 0, we have a pristine lattice, and with an angle of 90° we have a Stone-Wales defect. The error bars with the VDNNs come from using an ensemble of 5 VDNNs. The energy difference between the VDNNs and QMC is always less than the difference between DFT and QMC.

energy of the validation set. This total electronic energy excluded the ion-ion energy. In this case, the validation set was comprised of the atomic configurations with $\theta = 0$ and $\theta = \pi/2$. This rigid shift was then applied to the remaining 8 atomic configurations, which made up the testing set. In Figure 11.3a, we plot the energy barrier computed with the VDNNs, as well as the energy barriers found with DFT and QMC. For all values along this curve, the VDNN has a smaller absolute error than DFT and all energies are within chemical accuracy when compared to the QMC energies. In addition, all but 3 configurations have overlapping statistical error bars when comparing the energies computed with the VDNNs and QMC. For all methods, the largest value of energy occurs at $\theta = 50°$. We define the difference in energy between this configuration and the configuration with $\theta = 0$ to be the energy barrier needed to be overcome to transition from a pristine lattice to one with a

Stone-Wales defect. We find the energy barrier to be 55 meV / electron based on the QMC calculation. DFT underestimates the energy barrier by 4.4 meV / electron and VDNNs overestimates the energy barrier by 10 meV / electron. However, both of these estimates are within chemical accuracy (43.4 meV). Looking to Figure 11.3b, we remark an increase in error for VDNNs from 30-80°. Upon visual inspection of the true energy densities ($\mathcal{V}_{ee}$ and $\mathcal{V}_{ei}$) in Figure 11.2, we find that they spread uniformly away from the defect across the pristine lattice. However, the densities with random atomic perturbations do not have uniform energy spreading. Depending on the atomic perturbations, there are regions with high and low energy densities. The errors associated with the VDNN come directly from the prediction of the spreading of the energy density. Namely, with $\theta = 90°$, it overestimates the densities at the defect and underestimates the densities away from the defect. Although there are over/underestimates in the predictions, the VDNN is still capable of capturing this uniform spreading of the energy densities, despite not seeing similar densities during training.

Before concluding, we discuss limitations of VDNNs. Firstly, VDNNs are limited to the data they are trained with as is the case with all machine learning models. Secondly, for machine learning models that predict just scalar quantities (i.e. total energy), there is no error correction that must be done. However, as discussed previously, error from VDNNs accumulates when integrating, and must be accounted for. Thirdly, in VDNNs there is no dependence between neighbouring pixels that are output by the model. The dependence could be applied by outputting neighbouring pixels and averaging over the predictions or in some sort of post-processing routine. Fourthly, due to use of images, rotational invariance is not inherently built-in to the model, but must be learned during training. Lastly, when constructing diversity in a dataset, one must consider the structure of the electron density, rather than a chemical environment. The two are inter-linked, but one must consider how the electron density images change within a dataset, and use this information when applying it to a new system.

11.4 Conclusion

In conclusion, we have used VDNNs to map electron densities computed with DFT to energy densities calculated with QMC. We have demonstrated our methodology on graphene lattices with two tasks. In the first task, we showed that only 3 atomic configurations (QMC calculations) are needed to achieve chemical accuracy for graphene lattices with random displacements. In the second task, we studied the ability of VDNNs to reproduce an energy barrier curve for the transition of a pristine graphene lattice to a graphene lattice with a Stone-Wales defect. This transition was completed by rotating 2 atoms within the lattice. We find that the VDNNs are capable of accurately reproducing the energy densities along this transition, with a majority of the error associated with energy density spreading. In future work, algorithms to improve the generalization of VDNNs, such as active learning [181] could be used to detect unforeseen chemical environments. This could allow for a VDNN to generalize across different chemical motifs, phases, and materials, allowing for QMC energies at DFT computational cost. The datasets used in this manuscript can be found here [182].

11.5 Acknowledgements

The Authors acknowledge Compute Canada, the Vector Institute for Artificial Intelligence, and the National Energy Research Scientific Computing Center for computational resources. KR and IT acknowledges the National Sciences and Engineering Council of Canada for funding. KR acknowledges the Vector Institute for Artificial Intelligence for funding. Work at the National Research Council was carried out under the auspicious of the AI4D program.

11.A Supplemental Information

11.A.1 Jastrow Factors

The Jastrow factors were represented by B-splines and the coefficients were optimized using the quartic optimizer in QMCPACK [178]. The meshes used in the QMC calculations were increased by 20% and we found the average and corresponding standard deviation of the variance to energy ratio of 0.0205 ± 0.0014. The optimization was run for 100 blocks each consisting of 51200 Monte-Carlo samples.

11.A.2 Calculation of the Energy Densities

In this subsection, we provide additional details for the calculation of the energy densities. The first step is performing variational Monte Carlo (VMC) for 40 blocks with 30 warm-up steps and 10 steps used for averaging. Warm-up steps are steps taken where data is excluded from calculating quantities. The second step was DMC with a larger time step (0.02) for 20 blocks with 20 warm-up steps and 5 steps used for averaging. The final step was DMC with a smaller time step (0.01) for 200 blocks with 20 warm-up steps followed by 10 steps. The total number of Monte-Carlo samples completed for each of the final DMC calculations was 20 blocks $\times$ 10 steps $\times$ 1024 walkers $\times$ 64 MPI processes $\approx 13 \times 10^6$. The average and corresponding standard deviation of the total energy error across all calculations was 0.426 ± 0.083 meV.

11.A.3 Convergence Calculations

In Figure 11.4, we show the mean absolute error convergence as a function of number of training configurations.

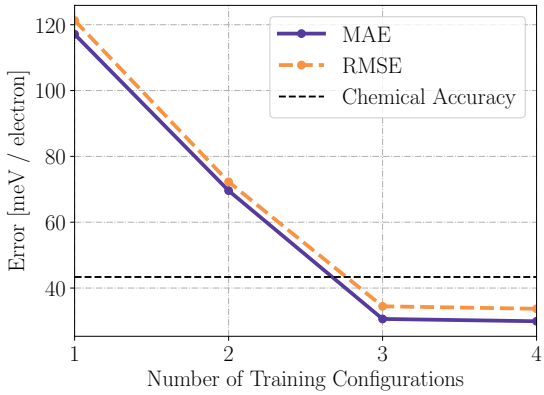


Figure 11.4: Mean absolute error and root mean absolute errors versus number of training configurations seen for pure graphene with random atomic perturbations. The ensemble models were averaged to compute the total energy. Convergence is seen with only 3 configurations.

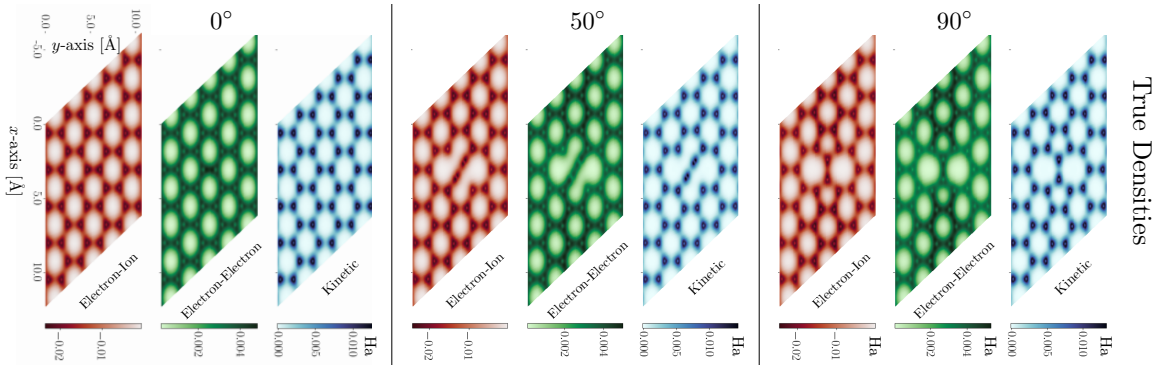


Figure 11.5: Energy densities of graphene computed with diffusion Monte Carlo where two atoms are rotated at a given angle.

11.A.4 Visualization of DMC Energy Densities

In Figure 11.5, we show the true energy densities of graphene that correspond with the predicted energy densities in the main manuscript.

Chapter 12

Inverse Design of a Graphene-Based Quantum Transducer via Neuroevolution

Thus far, we have focused on the acceleration of the computation of nanoscale materials. Afterwards, the model could be used within a design pipeline allowing one to traverse through a chemical space, enabling the discovery of a novel structure. In this chapter we focus on the design of a 2D material with applications to quantum computing. The electronic structure method used is tight binding, and therefore the computation of quantities is rapid. This methodological setup simulates an environment where one has access to a fast and scalable surrogate model. Due to this, we are able to traverse a large search space using an evolutionary algorithm. In addition, we introduce a methodology where a neural network policy behaves as a superoperator. This superoperator acts on the tight-binding Hamiltonian and returns a new Hamiltonian with properties that optimize the objective function.

Kevin Ryczko,
Department of Physics,
University of Ottawa

Pierre Darancet,
Argonne National Laboratory,

Isaac Tamblyn,
National Research Council of Canada,
Department of Physics, University of Ottawa

Abstract

We introduce an inverse design framework based on artificial neural networks, genetic algorithms, and tight-binding calculations, capable to optimize the very large configuration space of nanoelectronic devices. Our non-linear optimization procedure operates on trial Hamiltonians through superoperators controlling growth policies of regions of distinct doping. We demonstrate that our algorithm optimizes the doping of graphene-based three-terminal devices for valleytronics applications, monotonously converging to synthesizable devices with high merit functions in a few thousand evaluations (out of $\simeq 2^{3800}$ possible configurations). The best-performing device allowed for a terminal-specific separation of valley currents with $\simeq 96\%$ ($\simeq 94\%$) K (K') valley purity. Importantly, the devices found through our non-linear optimization procedure have both higher merit function and higher robustness to defects than the ones obtained through geometry optimization.

12.1 Introduction

Tailoring the properties of nanoelectronics devices leveraging quantum phenomena without classical analog is central to spintronics [183], valleytronics [184], quantum transduction, quantum sensing, and quantum computing [185, 186, 187, 188]. The figure of merit of such nanoelectronics devices strongly depends on the details of the low-energy Hamiltonian and can be impacted by numerous energy scales. For example, in the case of graphene-based devices [189], valley-polarized currents can be manipulated spatially by tuning the edges of the devices [190, 191], through strain-engineering [192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204], defect-engineering [205, 206], and by the combination of edge termination and doping via a gate voltage or molecular adsorption [190, 207, 208, 209, 210, 211]. While these studies predict high valley-filtering (e.g. with valley purity exceeding $\sim 95\%$ [190]), actual, synthesizable, devices will display a superposition of these effects, resulting in a complex optimization process of these possibly-competing energy scales.

Such optimization is an inverse design problem, where one knows the desired output of the device, as defined by a simple merit function (e.g. valley or spin purity, current magnitude, etc.), but does not know the optimal way to modify the device under experimental and synthesizability constraints to obtain local or global extrema of these merit functions. Due to their large tunability and the number of atoms they contain (typically exceeding 10^3), the configuration space of nanoelectronic devices is extremely large and impossible to fully explore with high-throughput forward quantum transport solvers [212, 213]. In this context, the coming-of-age of artificial intelligence approaches capable of optimizing in large configuration space [214] offers significant opportunities. Local optimization techniques were used, for example, to design physically-unintuitive demultiplexer devices [215, 216] and chemical compounds [217]. These approaches could also be accelerated by machine learning the quantities of interest, as demonstrated in Ref. [218].

Efficiently searching through large configuration spaces generally implies sampling a lower-dimensional space that maps to the original, higher dimensional, search space, possibly constructing this mapping with machine learning [219]. Another possibility is to use simple policies that yield complex outcomes, mimicking non-linear optimization processes

observed in nature. In the work of Schelling [220], the complex dynamics of segregation was modeled by subtle changes in the underlying policies. This was also shown to be the case by Wolfram [221] on cellular automata. The search of a large configuration space that is represented by an artificial neural network (ANN) can be done with GAs or gradient descent. GAs are gradient-free, are less likely to be trapped in local minima, and converge rapidly. On the contrary one must have a flexible ansatz (or set of genes) to fully describe the configuration space, and it is difficult to include a feasibility of fabrication score in the objective function. This could result in finding optimal structures with a GA that could not be fabricated in the laboratory. This issue is eliminated for generative machine learning models where gradients are used to minimize an objective function. Machine learning models are flexible with respect to input and ability to learn complex mappings, but one must initially train the model and be confident in the model's ability to perform inference over a wide range of structures.

In this work, we demonstrate inverse design of nanoelectronic devices using growth policies coupled to ANNs, GAs, and tight-binding calculations. The choice of GA was made because there is no analytic connection between the objective function and the policy network. Our framework is capable to navigate the very large configuration space through a non-linear optimization procedure that operates on trial Hamiltonians through a superoperator, effectively controlling growth policies of regions of distinct doping. We demonstrate optimization of the doping profile of graphene-based three-terminal devices for valleytronics applications, monotonously converging to synthesizable devices with high merit functions in a few thousand evaluations (out of $\simeq 2^{3800}$ possible configurations). The best-performing device allowed for a terminal-specific separation of valley currents with $\simeq 96\%$ ($\simeq 94\%$) K (K') valley purity. Importantly, the devices found through our non-linear optimization procedure have both higher merit function and higher robustness to defects than the ones obtained through linear geometrical optimization Section 12.A.

In the Methods section, we detail our theoretical approach which includes the structure design, optimization procedure, and the methodology for calculating valley polarized currents. In the Results section, we show that our optimization procedure produces structures that can separate valley polarized currents and analyze the GA optimization procedures.

12.2 Methods

Our general workflow is summarized in Fig. 12.1, and consists of a structure generation scheme using an ANN, a (forward) quantum transport solver, and a GA.

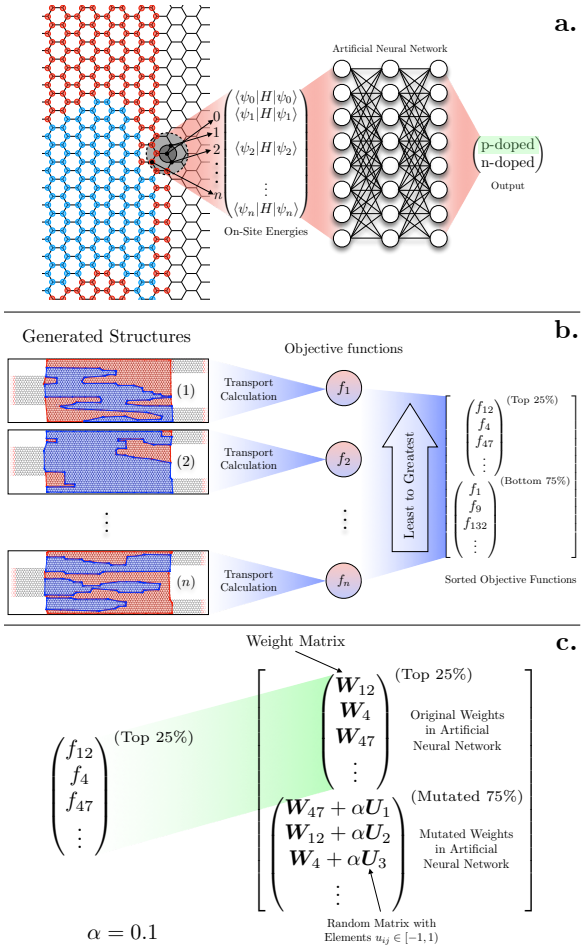


Figure 12.1: Schematic description of optimization procedure. Starting from a trial Hamiltonian (consisting of either p- or n-doped regions on a fixed device lattice) (a), an artificial neural network policy is used to modify the on-site energy based on the on-site energies of its neighboring sites (nearest and next-nearest neighbors). The on-site energies for this set of lattice sites are concatenated into an input vector which is fed into an ANN policy that determines whether the selected lattice site will be p- or n-doped. The resulting merit function of each trial structure is then computed with a tight-binding transport solver [212] (b). The structures shown were taken from a random optimization procedure, with regions of n- and p-doping highlighted for clarification. The structures are then ranked based on their objective functions. The structures in the next generation are produced (c) using a genetic algorithm on the policies of the top 25% of the performers in the population of previous generations.

Our framework operates on a fixed device structure, as represented by a graph Hamiltonian with a fixed adjacency matrix. Each vertex of the graph represents an orbital contributing to the low-energy Hamiltonian.

Each vertex has a corresponding on-site energy and a set of nearest neighbour couplings with sites defined in the adjacency matrix. The Hamiltonian of the system is

$$H = \sum_{\langle i,j \rangle} \tau_{ij} |i\rangle\langle j| + \sum_{i} U_i |i\rangle\langle i|, \tag{12.1}$$

where the first summation of Equation 12.1 describes the nearest neighbour interactions between orbitals i and j, and the second summation describes the on-site potential. The device is contacted to ballistic terminals with a predefined set of undoped lattice sites.

We focus on the case of three-terminal devices comprised of 3846 sites on a model of doped graphene, using $\tau_{ij} = -2.7$ eV as done in [189], and

$$U_i = \begin{cases} U & \text{n-doped} \\ -U & \text{p-doped} \\ 0 & \text{terminal} \end{cases} \tag{12.2}$$

with $U = 0.2$ eV, in alignment with past work [222]. This corresponds to a device with a body whose length is 128 Å. We also studied devices with shorter body lengths (down to 64 Å) and found no quantitative change in our results.

The spatial extent of the p- and n-doped doped regions on a given device is the result of an ANN policy that outputs the probability of given lattice sites to be p- or n-doped, as we now describe.

At the first iteration, we set all of the on-site energies to $U_i = 0.5$ eV. The choice of setting the on-site energies initially to 0.5 eV is arbitrary and the initialization can bias the optimization. We found that an initialization value < 0.5 biases K current and > 0.75 biases K' current. This bias is small for values close to the transition point but becomes non-negligible the further one moves from the transition point. For each lattice site i, the on-site energy along with the on-site energies of its nearest and next-nearest neighbors are concatenated into

an input vector $\mathbf{v}_{\text{input}}$. This choice of input vector is motivated by the structure of the nearest-neighbor tight-binding Hamiltonian.

This choice of input vector is equivalent to using a graph convolution layer [223, 65] where the weight matrix is shared amongst all atoms. The input vector $\mathbf{v}_{\text{input}}$ is then fed into the ANN policy where the output of the ANN is a number $p \in [0, 1]$. A random number $u_p \in [0, 1)$ is then drawn such that if $p < u_p$ the site is p-doped, and n-doped otherwise. This ANN policy can also be thought of as a superoperator that operates on a constant tight-binding Hamiltonian yielding a Hamiltonian with a set of desired optimized properties. At each iteration, the weights of the ANN policy are updated using a GA in conjunction with the ANN [224, 225].

To avoid producing structures where the doping changes over a short length scale, we apply Gaussian blurring as well as binary erosion and dilation on binary images of the lattice (1(0) represented p(n)-doping). Binary images were created by first initializing an array of zeros with dimensions equal to the length (128 Å) and width (64 Å) of the body. This made for an image with size 128×64. Afterwards, pixels were set to 1 if p-doped carbon atoms fall into a respective pixel. A Gaussian blur was then applied with a standard deviation of 2 Å, followed by a threshold operation with a value of 0.5, a binary erosion operation, and a binary dilation with a 50% chance. All of these operations can be written as kernels that operate, as a convolution, on the binary image. In addition, they can all be found in the scikit-image Python package [226]. We note that this post-processing protocol allowed for a 9% improvement in the objective function.

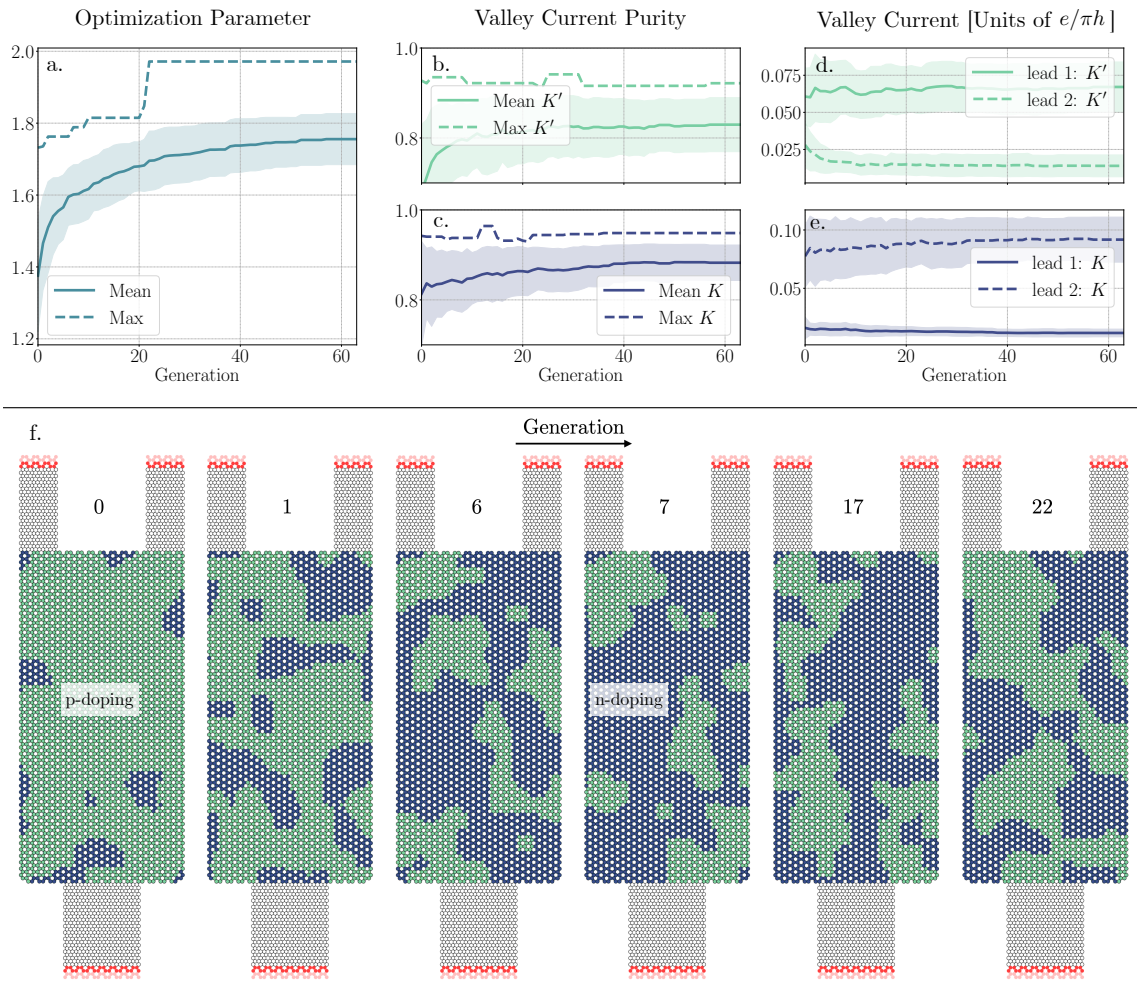


Figure 12.2: Optimization function (a), valley current purity (b-c), and valley currents (d-g) recorded during the optimization processes using the artificial neural network policy. For each independent optimization, the devices with the maximum objective functions are recorded. The solid lines show the average values of these devices and the lighter regions show the standard deviations. In f. we show the evolution of best performing structures for one seed.

The initialization stage of the optimization process consists of $N = 160$ devices, each corresponding to a distinct ANN policy.

Given these N Hamiltonians, we calculate the valley-polarized currents using the Kwant Python package [212], through the following equation:

$$I_{K/K'} = \frac{e}{\pi h} \int_{-\infty}^{\infty} d\epsilon \, G_{K/K'}[f(\epsilon; \mu_L) - f(\epsilon; \mu_R)] \tag{12.3}$$

where $G_{K/K'}$ is the valley-dependent conductance, $f(E; \mu_{L/R})$ is the Fermi-Dirac function

for the left (L) and right (R) leads, e is the electron charge, and h is Planck's constant. To calculate the valley-polarized conductance we use the Landauer-Büttiker formula

$$G_{K/K'} = \frac{2e^2}{h} \sum_{n_{\text{modes},K/K'}} T_n, \quad T_n = \sum_{m_{\text{modes}}} |t_{nm}|^2 \tag{12.4}$$

where the sum over $n_{\text{modes},K/K'}$ are for incoming modes with momenta associated with valley K or K', and the sum over m_{modes} are for all out going modes in the lead where we want to measure the current. Travelling modes (either K or K') are identified by considering their velocity and momentum, following [190]. The transmission probabilities T_n are computed after the determination of the scattering matrix. In addition, we use a bias of 0.5 eV and a grid spacing of 1 meV to evaluate the integral in Eq. 12.3 throughout all of our calculations.

We consider three-terminal devices with a non-valley-polarized current incoming from lead 0 (the "left" of the device in Fig. 12.1 b). At the opposite side of the device, two leads 1 and 2 at identical chemical potentials collect the valley-dependent current injected from lead 0.

Before defining the objective function, we first define the purity of K' current in lead 1 to be $P_{K',1} = I_{K',1}/(I_{K,1} + I_{K',1})$ and the purity of K current in lead 2 to be $P_{K,2} = I_{K,2}/(I_{K,2} + I_{K',2})$. We then define the normalized total current $I_{\text{total}} = (I_{K',1} + I_{K,2})/I_P$ where $I_P = 0.3\ e/\pi h$ is the total current of the pristine graphene lattice. We search for structures that maximize the multivariate objective function:

$$F(I_{K,1}, I_{K',1}, I_{K,2}, I_{K',2}) \quad = \quad P^2_{K',1} + P^2_{K,2} + I^2_{\text{total}}. \tag{12.5}$$

We compute Eq. 12.5 for each of the N devices. Only the ANN policies associated with the top 25% devices are kept to populate the next generation, through random mutation of the weights of the ANN policy yielding a new device. We do not consider crossover mutations in our study.

The weights of the newly generated ANN policy are

$$\begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{pmatrix}_{\text{new}} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{pmatrix}_{\text{old}} + \alpha \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_M \end{pmatrix} \tag{12.6}$$

where $\alpha = 0.1$ is a small parameter (similar to a learning rate), $u_i \in [-1, 1]$ is a random number, and M is the total number of weights. In our case we used ANNs with 2 hidden layers each with 128 neurons. The logistic activation function was used throughout the ANN. During the optimization process we ran calculations for 160 devices in parallel and performed 64 generations. The calculations performed for each generation took $\simeq 45$ minutes on single node with 40 CPUs.

12.3 Results

The results of the optimization procedure are shown in Figure 12.2. We show the average and maximum optimization function, valley current purity, and associated valley currents for the best performing devices of each independent seed as a function of generation following the Methods section. We also show the associated standard deviation over the $N = 64$ independent calculations. Both the averages of the objective function and purity of the valley currents converge monotonously across 64 generations. The optimization process produced a structure with maximum purity of 96% (94%) K (K') valley current. This result is comparable to Rycerz *et al.* [190] where they achieved $\sim 95\%$ purity with an idealized valley filtering device.

The average purity of K valley current at generation 0, based on the random initialization is higher ($> 80\%$) than the average purity of K' valley current for the best performing devices. In the remainder of the optimization process, the algorithm is capable to maintain the K valley current purity steady while increasing the value of the K' valley current purity, which started at an average value of $\sim 70\%$ purity.

In addition to the convergence of the average currents, average measures of purity, and the average of the objective functions we also find that the standard deviation of the optimizations decreases as a function of generation. This indicates that the population is converging to a similar local maxima.

In Figure 12.2 f, we also show the evolution of the best-performing structures for a single seed.

To further understand the devices resulting from the optimization procedures, we investigate the doping profiles and their effects on valley currents. In particular, we perform similar optimizations as discussed in the Methods section but only allow for either p- or n-doping, rather than both. In Figure 12.3 b-c we plot the likelihood that a given site would have p- or n-doping when the valley currents are optimized. The likelihood comes from averaging over the final structures from the 64 optimizations with different seeds. We see the algorithm prefers uniform doping directly in contact with the leads, with the p(n)-doping acting as a waveguide for the $K'(K)$ valley current towards lead 1 (2). Each choice of doping guides one of the valley currents and has little effect on the other valley current. In the case of p(n)-doping, the $K(K')$ valley current had a valley current purity of $\simeq 57\%$. In addition to these experiments, we also performed calculations to optimize either K or K' purity, allowing for both p- and n-doping, and following the same protocol as described in the Methods section. We found that we could reach 96% purity for K' and 97% purity for K, which is on-par with the valley purity reported previously [190].

In contrast with this doping-induced behavior for single valley polarization, devices optimized using Eq. 12.5 shows mixed doping in front of lead 0, and weak long-range order, as shown in Fig 12.3. To quantify to length scale of the ordering of the devices found by optimizing Equation 12.5, we calculated pair correlation functions (PCFs) $g(r)$ where the atom types are labelled by their respective doping. We applied Gaussian blurring with a standard deviation of 0.075 Å to the PCFs to eradicate the peaks from the regular lattice structure. If $(g(r)+1)/(g_{\mathrm{CC}}(r)+1) \approx 1$, then we expect to see uniform doping across the crystal. If $(g(r)+1)/(g_{\mathrm{CC}}(r)+1) \approx 0$, then we expect to see non-uniform doping across

the crystal. For $(g_{\mathrm{pp}}(r)+1)/(g_{\mathrm{CC}}(r)+1)$ and $(g_{\mathrm{nn}}(r)+1)/(g_{\mathrm{CC}}(r)+1)$, we find that the values decrease from 1 as r increases. Therefore if we have a site that is p(n)-doped, we expect adjacent sites to be p(n)-doped for up to 10 Å with high probability. Beyond 10 Å, there is a $\sim 50\%$ chance to see a p(n)-doped site given p(n)-doped site is selected, indicating seemingly random arrangement. The slight discrepancy between $g_{pp}(r)$ and $g_{nn}(r)$ beyond 10 Å in Figure 12.3a is due to more sites being n-doped.

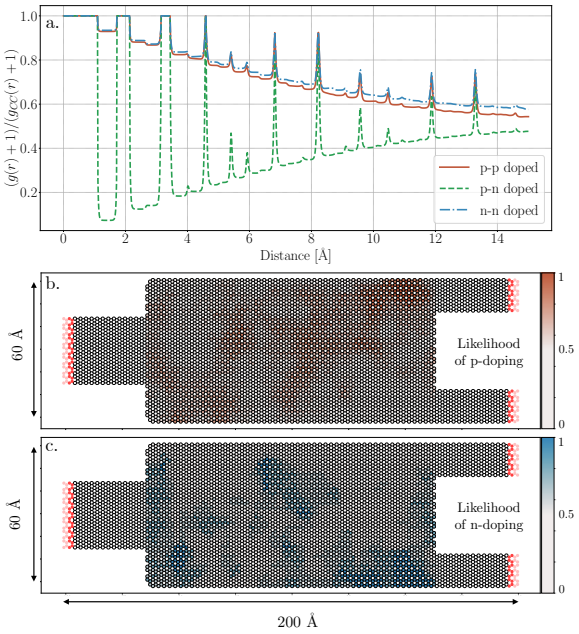


Figure 12.3: Pair correlation functions (PCFs - $g(r)+1$) divided by the PCF of graphene ($g_{\mathrm{CC}}(r)+1$) where p- and n-doping are considered to be different atoms (a). Likelihood of finding a given lattice site with p-doping (b) and likelihood of finding a given lattice site with n-doping (c) for the optimizations where sites were either p(n)-doped or undoped.

We note that this 10 Å length scale exceeds the information from neighbors and next-nearest neighbors given to the superoperator, and are a result of the ANN policy network and blurring protocol; the local environment encoded in the input vectors overlap allowing information to be propagated. Without the blurring protocol this length scale drops to 6 Å but still exceeds the distance between a site and it's next-nearest neighbour. The same effect can be seen when machine learning potential energy surfaces using fingerprint functions to describe atomic environments [11, 19]. These fingerprint functions have a cut-off radius that may be smaller than a certain interaction distance (i.e. vdW interactions), but because of the overlap of the fingerprints, they still can describe interactions that exceed

the cut-off radius.

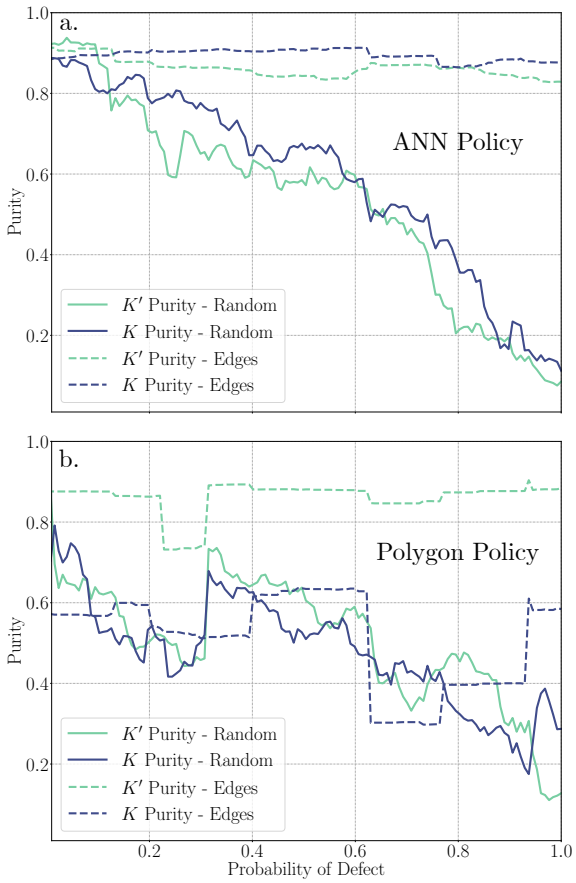


Figure 12.4: Purity of the most optimal device as a function of the probability that a lattice site will have a defect (flipped doping). Solid curves are for randomly selected sites and dashed curves are for sites separating p-doped regions from n-doped regions (edge sites). The top plot is for the ANN policy (a), and the bottom plot is for the polygon policy (b). See the Methods section for more information on the ANN policy and the supplemental information Section 12.A for more information on the Polygon policy.

Lastly, we investigate the sensitivity of the generated structure that had the maximum objective function. To do so, we consider two protocols. In the first protocol that we refer to as the 'random' protocol, we scan through every lattice site and flip the doping (p- to n-doping or vice versa) of the site if $u < p$ where u is a randomly generated number $u \in [0, 1)$ and p is the probability that the doping will be flipped. In the second protocol, we only consider flipping the doping of lattice sites that separate p-doped regions from n-doped regions. We refer to these lattice sites as 'edges' and refer to this protocol as the 'edge' protocol. One should note that the number of edge sites makes up a small fraction of the total number of sites in the device.

In Figure 12.4, we show the purity of the valley currents as a function of the probability p for the ANN policy outlined in the Methods section as well as our polygon policy outlined in the supplemental information Section 12.A. For the ANN policy with the random protocol, we find that the purity of both K and K' valley currents decrease linearly (with noise) as a function of the probability p. This is in agreement with [190], where random vacancies were introduced in the lattice. For the ANN policy with the edge protocol, we find that the purity remains almost constant, with a slight decay as the probability p increases. This indicates that the proposed structure is robust to changes around the edges of p- or n-doped regions.

In contrast, for the polygon policy with the random protocol we find a decrease of the purity of valley currents as a function of the probability p, but with large steps at certain values of p. These large jumps in purity are also observed for the polygon policy with the edge protocol. These large and random jumps in the purity indicate the sensitivity of this protocol. Electron waves can be focused and split, similar to light waves, depending on the shape of the doped regions [207, 209]. When one changes the curvature of a lens slightly, the behavior of light can be drastically different. A similar process is occurring here with the electron waves.

12.4 Conclusion

In conclusion, we describe a technique that uses genetic algorithms and artificial neural network policies to optimize the purity of valley currents in graphene nanodevices with pn-doping. This optimization strategy operates on a tight-binding Hamiltonian and yields a new, optimized Hamiltonian for our objective function. This technique allows for rapid convergence of the optimization parameter studied, and yields similar solutions from independent calculations with different seeds. After averaging over an ensemble of optimization procedures we have found that p(n)-doping acts as a waveguide for $K'(K)$ valley current, allowing one to physically separate valley currents in graphene nanoribbons. After averaging over the ensemble of optimization procedures with both p- and n-doping, we found that the purity of the valley currents were $\simeq 93\%$. The best-performing device allowed for a terminal-specific separation of valley currents with $\simeq 96\%$ ($\simeq 94\%$) K (K') valley purity.

When averaging over the ensemble of optimization procedures with only p(n)-doping, we found that the purity of $K'(K)$ valley remains at $\simeq 93\%$. This shows that p(n)-doping acts as a guide to $K'(K)$ valley current. We also achieve a valley purity of 96% for K' and 97% for K current when only optimizing one valley. Additionally, we found that the artificial neural network policy can produce structures with long-range order despite only having local information. We also performed sensitivity analysis which showed that the proposed optimal structure of the artificial neural network policy is robust to edge defects. Such a device could be used to convert a quantum state to a digital signal.

12.5 Acknowledgements

KR and IT acknowledge funding from the Natural Sciences and Engineering Research Council of Canada. Work at the National Research Council was carried out under the auspices of the AI4Design Program. KR and IT acknowledge Compute Canada and the National Energy Research Scientific Computing Center for computational resources. This material is based upon work supported by Laboratory Directed Research and Development (LDRD) funding from Argonne National Laboratory, provided by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-06CH11357. Use of the Center for Nanoscale Materials, an Office of Science user facility, was supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-06CH11357.

12.A Supplemental Information

12.A.1 A Proof of Principle for Device Optimization

In what we call our 'polygon strategy' we define a n-doped region by defining a set of 6 points and drawing a convex hull. All other regions of the device are p-doped. Such a device can be seen in Figure 12.5. The genes of the structure are the set of points that defines the convex hull. The values y_1, y_2, y_4, and y_5 are fixed during the optimization process.

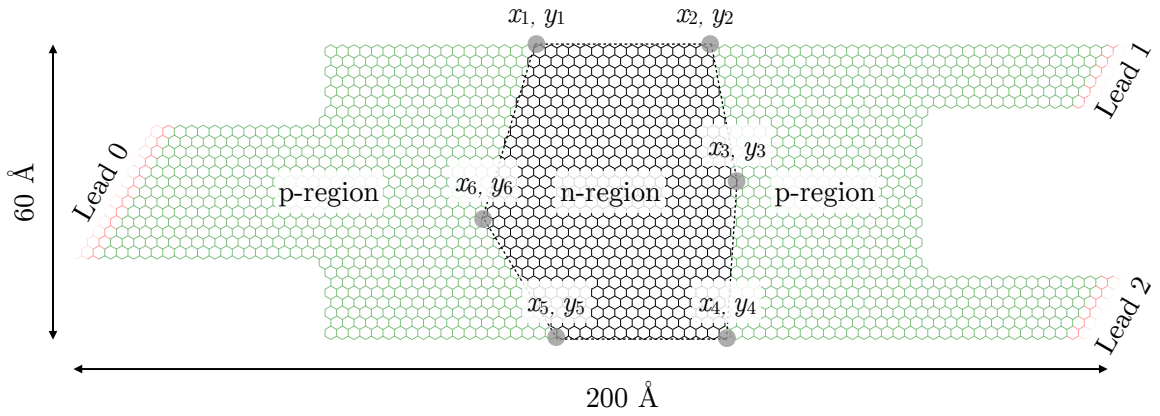


Figure 12.5: Visual representation of the polygon strategy. The lattice sites that are within a polygon that is drawn with 6 points are declared to be p-doped where everything else is declared to be n-doped.

The same methodology outlined in the Methods section of the manuscript is applied here, except the set of points are updated rather than the weights of the artificial neural network (ANN) policy:

$$\begin{pmatrix} x_1, y_1 \\ x_2, y_2 \\ \vdots \\ x_6, y_6 \end{pmatrix}_{\text{new}} = \begin{pmatrix} x_1, y_1 \\ x_2, y_2 \\ \vdots \\ x_6, y_6 \end{pmatrix}_{\text{old}} + \alpha \begin{pmatrix} u_{11}, u_{12} \\ u_{21}, u_{22} \\ \vdots \\ u_{61}, u_{62} \end{pmatrix} \tag{12.7}$$

Following the same protocol as before we ran 64 independent optimizations, each with a different random seed. In each of the optimizations, we track the structure that maximizes the optimization function. In Figure 12.6 we plot the average, maximum and standard deviation optimization function, purity of valley currents, and valley currents of those devices. Comparing to the results of the ANN policy, we can see that the performance in this strategy is weaker. We obtain an average valley current purity of $\sim 82\%$ for K' and $\sim 78\%$ for K.

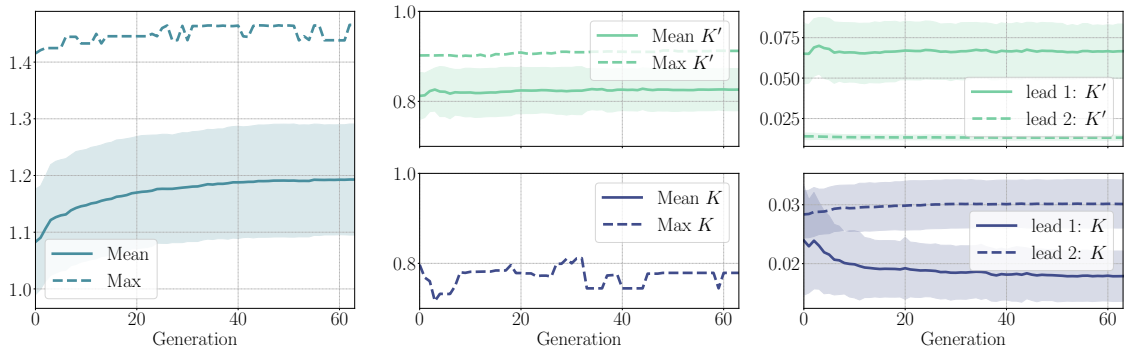


Figure 12.6: Optimization curves (left), valley current purity (middle), and valley currents (right) recorded during the optimization processes using the polygon policy. For each independent optimization, the devices with the maximum objective functions are recorded. The solid lines show the average values of these devices and the lighter regions show the standard deviations.

Part IV

Concluding Remarks

12.2 Summary

This thesis contained three parts which we now summarize. In Part I, we introduced and discussed concepts of computation and design when applied to nanoscale materials. This first included discussion of how previous work has approached the computation and design of atomistic systems. From this discussion we identified that the use of featureless machine learning approaches, where one operates on scalar fields and uses image-based deep learning technologies developed by the machine learning community. were absent in the literature. This is the approach that was taken throughout this thesis. Additionally, the field of inverse design is emerging and open for contribution. In Part II, we then presented relevant theories used to accomplish the work that contributed to the articles included in this thesis. This included electronic structure and density functional theory (DFT), density functional perturbation theory (DFPT), quantum Monte Carlo (QMC), machine and deep learning, and electron transport with tight binding. In Part III, we presented 5 articles that contributed to the acceleration of computation of nanoscale materials (which enable design) and 1 article that contributed to the design. For the computation, in Chapter 7 we began our efforts using a simple image-based representation of an atomistic system. This allowed us to leverage deep convolutional neural networks (CNNs) to make accurate energy predictions of various systems. However, this methodology required large datasets and was not extensive. The trained model could not be used to perform additional studies on similar systems with differing size. In Chapter 8, we utilized extensive deep neural networks (EDNNs) on 2D model systems, where we showed that these networks are capable of cooperative learning; total internal energy contributions were predicted simultaneously. In addition, we used deep convolutional inverse graphics networks to map scalar fields rather than a scalar field to a scalar value. EDNNs allow one to train on smaller systems and perform inference on larger ones, but there is still an upfront cost of generating a large dataset. Both of these limitations were addressed in Chapter 9, where voxel DNNs (VDNNs) were introduced and utilized for kinetic energy density and functional derivative of the kinetic energy predictions. With VDNNs, only a handful of calculations are needed, since one predicts an energy density rather than the integrated energy density. In addition, one

can study any system size as long as the grid spacing is constant. This methodology was also applied in Chapter 11, where electron densities computed via density functional theory were mapped to energy densities computed via diffusion Monte Carlo (DMC). Again, it was shown that only a handful of calculations are needed to make accurate DMC predictions. Existing machine learning methods require large datasets that consist of at least $\mathcal{O}(10^3)$ calculations. DMC calculations are very computationally demanding and therefore one can not construct such a dataset with ease. VDNNs show great promise when applied to this scenario. In Chapter 10, we focused on accelerating the computation of response quantities rather than just ground state and developed an image-based DNN architecture with built in extensivity. This architecture combines the atom-centered approach taken in past literature along with the image-based approach taken in this thesis. In the final article (Chapter 12), we focus on the inverse design of a graphene lattice, such that valley currents can be physically separated. To do so, we used a genetic algorithm, a neural network policy network, and tight binding calculations. Using this approach, we were able to rapidly converge to devices where valley current purity was $\approx 95\%$.

12.3 Outlook

As machine learning methods applied to nanoscale systems continue to mature and become more transferable to systems that vary greatly in chemical space, the more useful they will be for inverse design. The ultimate goal of computational material science is to aid in understanding the properties of materials, such that we can design new materials with desired properties. Therefore, novel methods constructed to accelerate traditional computations alone are only useful when they can be applied to aid in the understanding of current or new materials. A majority of this thesis has been dedicated to the acceleration of the computation of nanoscale materials via machine learning. However, this is also the focus of many research groups around the globe. Therefore, it is wise to consider how best to apply these machine learning techniques to design problems. The future research for the present author will focus on the transferability of such approaches and applying these approaches to different design problems. It is the combination of these two research domains that will

ultimately lead to the discovery of novel materials that will allow us to have a successful and sustainable future on this planet.

APPENDIX A

Reproduction Rights

A.1 Convolutional Neural Networks for Atomistic Systems

"As the author of this Elsevier article, you retain the right to include it in a thesis or dissertation, provided it is not published commercially. Permission is not required, but please ensure that you reference the journal as the original source."

A.2 Extensive deep neural networks for transferring small scale learning to large scale systems

A.3 Deep Learning and Density Functional Theory

American Physical Society Reuse and Permissions License

29-Nov-2021

This license agreement between the American Physical Society ("APS") and Kevin Ryczko ("You") consists of your license details and the terms and conditions provided by the American Physical Society and SciPris.

Licensed Content Information

License Number:	**RNP/21/NOV/047315**
License date:	29-Nov-2021
DOI:	10.1103/PhysRevA.100.022512
Title:	Deep learning and density-functional theory
Author:	Kevin Ryczko, David A. Strubbe, and Isaac Tamblyn
Publication:	Physical Review A
Publisher:	American Physical Society
Cost:	USD $ 0.00

Request Details

Does your reuse require significant modifications: No

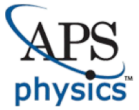
Specify intended distribution locations:	Canada
Reuse Category:	Reuse in a thesis/dissertation
Requestor Type:	Author of requested content
Items for Reuse:	Whole Article
Format for Reuse:	Electronic

Information about New Publication:

University/Publisher:	
Title of dissertation/thesis:	Accelerating the Computation and Design of Nanoscale Materials with Deep Learning
Author(s):	
Expected completion date:	Dec. 2021

License Requestor Information

Name:	Kevin Ryczko
Affiliation:	Individual
Country:	Canada



American Physical Society
Reuse and Permissions License

TERMS AND CONDITIONS

The American Physical Society (APS) is pleased to grant the Requestor of this license a non-exclusive, non-transferable permission, limited to Electronic format, provided all criteria outlined below are followed.

1. You must also obtain permission from at least one of the lead authors for each separate work, if you haven't done so already. The author's name and affiliation can be found on the first page of the published Article.
2. For electronic format permissions, Requestor agrees to provide a hyperlink from the reprinted APS material using the source material's DOI on the web page where the work appears. The hyperlink should use the standard DOI resolution URL, http://dx.doi.org/{DOI}. The hyperlink may be embedded in the copyright credit line.
3. For print format permissions, Requestor agrees to print the required copyright credit line on the first page where the material appears: "Reprinted (abstract/excerpt/figure) with permission from [(FULL REFERENCE CITATION) as follows: Author's Names, APS Journal Title, Volume Number, Page Number and Year of Publication.] Copyright (YEAR) by the American Physical Society."
4. Permission granted in this license is for a one-time use and does not include permission for any future editions, updates, databases, formats or other matters. Permission must be sought for any additional use.
5. Use of the material does not and must not imply any endorsement by APS.
6. APS does not imply, purport or intend to grant permission to reuse materials to which it does not hold copyright. It is the requestor's sole responsibility to ensure the licensed material is original to APS and does not contain the copyright of another entity, and that the copyright notice of the figure, photograph, cover or table does not indicate it was reprinted by APS with permission from another source.
7. The permission granted herein is personal to the Requestor for the use specified and is not transferable or assignable without express written permission of APS. This license may not be amended except in writing by APS.
8. You may not alter, edit or modify the material in any manner.
9. You may translate the materials only when translation rights have been granted.
10. APS is not responsible for any errors or omissions due to translation.
11. You may not use the material for promotional, sales, advertising or marketing purposes.
12. The foregoing license shall not take effect unless and until APS or its agent, Aptara, receives payment in full in accordance with Aptara Billing and Payment Terms and Conditions, which are incorporated herein by reference.
13. Should the terms of this license be violated at any time, APS or Aptara may revoke the license with no refund to you and seek relief to the fullest extent of the laws of the USA. Official written notice will be made using the contact information provided with the permission request. Failure to receive such notice will not nullify revocation of the permission.
14. APS reserves all rights not specifically granted herein.
15. This document, including the Aptara Billing and Payment Terms and Conditions, shall be the entire agreement between the parties relating to the subject matter hereof.

A.4 Inverse Design of a Graphene-Based Quantum Transducer via Neuroevolution

Inverse Design of a Graphene-Based Quantum Transducer via Neuroevolution

Author: Kevin Ryczko, Pierre Darancet, Isaac Tamblyn
Publication: The Journal of Physical Chemistry C
Publisher: American Chemical Society
Date: Dec 1, 2020

References

[1] TMI Mahlia, TJ Saktisahdan, A Jannifar, MH Hasan, and HSC Matseelar. A review of available methods and development on energy storage; technology update. *Renewable and Sustainable Energy Reviews*, 33:532–545, 2014.

[2] Peter Csermely, Tamás Korcsmáros, Huba JM Kiss, Gábor London, and Ruth Nussinov. Structure and dynamics of molecular networks: a novel paradigm of drug discovery: a comprehensive review. *Pharmacology & Therapeutics*, 138(3):333–408, 2013.

[3] Jae Sang Heo, Jimi Eom, Yong-Hoon Kim, and Sung Kyu Park. Recent progress of textile-based wearable electronics: a comprehensive review of materials, devices, and applications. *Small*, 14(3):1703034, 2018.

[4] Oluwafunmilola Ola and M Mercedes Maroto-Valer. Review of material design and reactor engineering on tio2 photocatalysis for co2 reduction. *Journal of Photochemistry and Photobiology C: Photochemistry Reviews*, 24:16–42, 2015.

[5] Gabriel R Schleder, Antonio CM Padilha, Carlos Mera Acosta, Marcio Costa, and Adalberto Fazzio. From dft to machine learning: recent approaches to materials science–a review. *Journal of Physics: Materials*, 2(3):032001, 2019.

[6] S Marchini, S Günther, and J Wintterlin. Scanning tunneling microscopy of graphene on ru (0001). *Physical Review B*, 76(7):075429, 2007.

[7] Timothy C Germann. Large-scale classical molecular dynamics simulations of shock-induced plasticity in bcc niobium. In *AIP conference proceedings*, volume 1195, pages 761–764. American Institute of Physics, 2009.

[8] Richard M Martin. *Electronic structure: basic theory and practical methods*. Cambridge University Press, 2020.

[9] RA Johnson. Empirical potentials and their use in the calculation of energies of point defects in metals. *Journal of Physics F: Metal Physics*, 3(2):295, 1973.

[10] Frank Noé, Alexandre Tkatchenko, Klaus-Robert Müller, and Cecilia Clementi. Machine learning for molecular simulation. *Annual Review of Physical Chemistry*, 71(1):361–390, 2020. PMID: 32092281.

[11] Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Physical Review Letters*, 98(14):146401, 2007.

[12] Albert P Bartók, Mike C Payne, Risi Kondor, and Gábor Csányi. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Physical Review Letters*, 104(13):136403, 2010.

[13] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O Anatole Von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical Review Letters*, 108(5):058301, 2012.

[14] John C Snyder, Matthias Rupp, Katja Hansen, Klaus-Robert Müller, and Kieron Burke. Finding density functionals with machine learning. *Physical Review Letters*, 108(25):253002, 2012.

[15] Albert P Bartók, Risi Kondor, and Gábor Csányi. On representing chemical environments. *Physical Review B*, 87(18):184115, 2013.

[16] Kristof T Schütt, Farhad Arbabzadah, Stefan Chmiela, Klaus R Müller, and Alexandre Tkatchenko. Quantum-chemical insights from deep tensor neural networks. *Nature Communications*, 8:13890, 2017.

[17] Felix Brockherde, Leslie Vogt, Li Li, Mark E Tuckerman, Kieron Burke, and Klaus-Robert Müller. Bypassing the kohn-sham equations with machine learning. *Nature Communications*, 8(1):1–10, 2017.

[18] Stefan Chmiela, Alexandre Tkatchenko, Huziel E Sauceda, Igor Poltavsky, Kristof T Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving

molecular force fields. *Science Advances*, 3(5):e1603015, 2017.

[19] Kristof T Schütt, Huziel E Sauceda, P-J Kindermans, Alexandre Tkatchenko, and K-R Müller. Schnet–a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722, 2018.

[20] Kevin Ryczko, Kyle Mills, Iryna Luchak, Christa Homenick, and Isaac Tamblyn. Convolutional neural networks for atomistic systems. *Computational Materials Science*, 149:134–142, 2018.

[21] Zhuoran Qiao, Matthew Welborn, Animashree Anandkumar, Frederick R Manby, and Thomas F Miller III. Orbnet: Deep learning for quantum chemistry using symmetry-adapted atomic-orbital features. *The Journal of Chemical Physics*, 153(12):124111, 2020.

[22] Michael Gastegger, Kristof T Schütt, and Klaus-Robert Müller. Machine learning of solvent effects on molecular spectra and reactions. *Chemical Science*, 12(34):11473–11483, 2021.

[23] Johannes Klicpera, Shankari Giri, Johannes T Margraf, and Stephan Günnemann. Fast and uncertainty-aware directional message passing for non-equilibrium molecules. *arXiv preprint arXiv:2011.14115*, 2020.

[24] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole von Lilienfeld. Big data meets quantum chemistry approximations: the Δ-machine learning approach. *Journal of Chemical Theory and Computation*, 11(5):2087–2096, 2015.

[25] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Advances in Neural Information Processing Systems*, pages 991–1001, 2017.

[26] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

[27] Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, et al. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL materials*, 1(1):011002, 2013.

[28] Anand Chandrasekaran, Deepak Kamal, Rohit Batra, Chiho Kim, Lihua Chen, and Rampi Ramprasad. Solving the electronic structure problem with machine learning. *NPJ Computational Materials*, 5(1):1–7, 2019.

[29] Ralf Meyer, Manuel Weichselbaum, and Andreas W Hauser. Machine learning approaches toward orbital-free density functional theory: Simultaneous training on the kinetic energy density functional and its functional derivative. *Journal of Chemical Theory and Computation*, 16(9):5685–5694, 2020.

[30] Yi Zhou, Jiang Wu, Shuguang Chen, and GuanHua Chen. Toward the exact exchange–correlation potential: A three-dimensional convolutional neural network construct. *The Journal of Physical Chemistry Letters*, 10(22):7264–7269, 2019.

[31] O Anatole von Lilienfeld. Towards the computational design of compounds from first principles. In *Many-Electron Approaches in Physics, Chemistry and Mathematics*, pages 169–189. Springer, 2014.

[32] Andriy Zakutayev, Xiuwen Zhang, Arpun Nagaraja, Liping Yu, Stephan Lany, Thomas O Mason, David S Ginley, and Alex Zunger. Theoretical prediction and experimental realization of new stable inorganic materials using the inverse design approach. *Journal of the American Chemical Society*, 135(27):10048–10054, 2013.

[33] Yue-Yu Zhang, Weiguo Gao, Shiyou Chen, Hongjun Xiang, and Xin-Gao Gong. Inverse design of materials by multi-objective differential evolution. *Computational Materials Science*, 98:51–55, 2015.

[34] Juhwan Noh, Jaehoon Kim, Helge S Stein, Benjamin Sanchez-Lengeling, John M Gregoire, Alan Aspuru-Guzik, and Yousung Jung. Inverse design of solid-state materials via a continuous representation. *Matter*, 1(5):1370–1384, 2019.

[35] Zhenpeng Yao, Benjamín Sánchez-Lengeling, N Scott Bobbitt, Benjamin J Bucior, Sai Govind Hari Kumar, Sean P Collins, Thomas Burns, Tom K Woo, Omar K Farha, Randall Q Snurr, et al. Inverse design of nanoporous crystalline reticular materials with deep generative models. *Nature Machine Intelligence*, 3(1):76–86, 2021.

[36] Kyungdoc Kim, Seokho Kang, Jiho Yoo, Youngchun Kwon, Youngmin Nam, Dongseon Lee, Inkoo Kim, Youn-Suk Choi, Yongsik Jung, Sangmo Kim, et al. Deep-learning-based inverse design model for intelligent discovery of organic molecules. *NPJ Computational Materials*, 4(1):1–7, 2018.

[37] TE Sharp. Potential-energy curves for molecular hydrogen and its ions. *Atomic Data and Nuclear Data Tables*, 2:119–169, 1970.

[38] Llewellyn H Thomas. The calculation of atomic fields. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 23, pages 542–548. Cambridge University Press, 1927.

[39] E Fermi. Atti rentes accad. *Naz. Lincei Rend. Cl. Sci. Fis, Mat. Natur*, 6:602, 1927.

[40] Pierre Hohenberg and Walter Kohn. Inhomogeneous electron gas. *Physical Review*, 136(3B):B864, 1964.

[41] Walter Kohn and Lu Jeu Sham. Self-consistent equations including exchange and correlation effects. *Physical Review*, 140(4A):A1133, 1965.

[42] John C Slater. A simplification of the hartree-fock method. *Physical Review*, 81(3):385, 1951.

[43] David M Ceperley and Berni J Alder. Ground state of the electron gas by a stochastic method. *Physical Review Letters*, 45(7):566, 1980.

[44] John P Perdew, Kieron Burke, and Matthias Ernzerhof. Generalized gradient approximation made simple. *Physical Review Letters*, 77(18):3865, 1996.

[45] Axel D Becke. Density-functional thermochemistry. iii. the role of exact exchange. *The Journal of Chemical Physics*, 98(7):5648–5652, 1993.

[46] Stefano Baroni, Stefano De Gironcoli, Andrea Dal Corso, and Paolo Giannozzi. Phonons and related crystal properties from density-functional perturbation theory. *Reviews of Modern Physics*, 73(2):515, 2001.

[47] Raffaele Resta and David Vanderbilt. Theory of polarization: a modern approach. In *Physics of Ferroelectrics*, pages 31–68. Springer, 2007.

[48] Xavier Gonze and J-P Vigneron. Density-functional approach to nonlinear-response coefficients of solids. *Physical Review B*, 39(18):13120, 1989.

[49] David Snoke. *Solid state physics: Essential concepts*. Cambridge University Press, 2020.

[50] Georg Placzek. *Rayleigh-streuung und Raman-effekt*, volume 2. Akademische Verlagsgesellschaft, 1934.

[51] SA Prosandeev, U Waghmare, Igor Levin, and J Maslar. First-order raman spectra of a b 1/ 2' b 1/ 2 "o 3 double perovskites. *Physical Review B*, 71(21):214307, 2005.

[52] M Veithen, Xavier Gonze, and Ph Ghosez. Nonlinear optical susceptibilities, raman efficiencies, and electro-optic tensors from first-principles density functional perturbation theory. *Physical Review B*, 71(12):125107, 2005.

[53] WMC Foulkes, Lubos Mitas, RJ Needs, and Guna Rajagopal. Quantum monte carlo simulations of solids. *Reviews of Modern Physics*, 73(1):33, 2001.

[54] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[55] D Ceperley. Ground state of the fermion one-component plasma: A monte carlo study in two and three dimensions. *Physical Review B*, 18(7):3126, 1978.

[56] Yang You, Igor Gitman, and Boris Ginsburg. Scaling sgd batch size to 32k for imagenet training. *arXiv preprint arXiv:1708.03888*, 6:12, 2017.

[57] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[58] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[59] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[60] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[61] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[62] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[63] Kyle Mills, Michael Spanner, and Isaac Tamblyn. Deep learning and the schrödinger equation. *Physical Review A*, 96(4):042113, 2017.

[64] Oliver T Unke, Stefan Chmiela, Huziel E Sauceda, Michael Gastegger, Igor Poltavsky, Kristof T Schutt, Alexandre Tkatchenko, and Klaus-Robert Muller. Machine learning force fields. *Chemical Reviews*, 2021.

[65] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[66] Supriyo Datta. *Electronic transport in mesoscopic systems*. Cambridge University Press, 1997.

[67] Geert Brocks. Electron transport at the nanoscale lecture notes, preliminary version, 2005.

[68] M. Born and R. Oppenheimer. Zur Quantentheorie der Molekeln. *Annalen der Physik*, 389(20):457–484, 1927.

[69] W. Kohn and L. J. Sham. Self-consistent equations including exchange and correlation effects. *Physical Review*, 140(4A), 1965.

[70] Laura E. Ratcliff, Stephan Mohr, Georg Huhs, Thierry Deutsch, Michel Masella, and Luigi Genovese. Challenges in Large Scale Quantum Mechanical Calculations. *WIREs Computational Molecular Science*, 7(1):e1290, jan 2016.

[71] Rocio Mercado, Bess Vlaisavljevich, Li-Chiang Lin, Kyuho Lee, Yongjin Lee, Jarad A. Mason, Dianne J. Xiao, Miguel I. Gonzalez, Matthew T. Kapelewski, Jeffrey B. Neaton, and Berend Smit. Force Field Development from Periodic Density Functional Theory Calculations for Gas Separation Applications Using MetalâĂŞOrganic Frameworks. *The Journal of Physical Chemistry C*, 120(23):12590–12604, 2016.

[72] Mathew J. Cherukara, Badri Narayanan, Alper Kinaci, Kiran Sasikumar, Stephen K. Gray, Maria K.Y. Chan, and Subramanian K R S Sankaranarayanan. Ab Initio - Based Bond Order Potential to Investigate Low Thermal Conductivity of Stanene Nanostructures. *The Journal of Physical Chemistry Letters*, 7(19):3752–3759, oct 2016.

[73] Ambarish R. Kulkarni and David S. Sholl. DFT-Derived Force Fields for Modeling Hydrocarbon Adsorption in MIL-47(V). *Langmuir*, 31(30):8453–8468, 2015.

[74] Andres Jaramillo-Botero, Saber Naserifar, and William A. Goddard. General multiobjective force field optimization framework, with application to reactive force fields for silicon carbide. *Journal of Chemical Theory and Computation*, 10(4):1426–1439, 2014.

[75] B. W H Van Beest, G. J. Kramer, and R. A. Van Santen. Force fields for silicas and aluminophosphates based on ab initio calculations. *Physical Review Letters*, 64(16):1955–1958, 1990.

[76] Jay W. Ponder and David A. Case. Force fields for protein simulations. *Advances in Protein Chemistry*, 66:27–85, 2003.

[77] Viktor Hornak, Robert Abel, Asim Okur, Bentley Strockbine, Adrian Roitberg, and Carlos Simmerling. Comparison of multiple amber force fields and development of improved protein backbone parameters. *Proteins: Structure, Function and Genetics*, 65(3):712–725, nov 2006.

[78] van der Waals. *De continuiteit van den gasen Vloeistoftoestand*. PhD thesis, Leiden University, 1873.

[79] Alberto Pérez, Iván Marchán, Daniel Svozil, Jiri Sponer, Thomas E Cheatham, Charles a Laughton, and Modesto Orozco. Refinement of the AMBER force field for nucleic acids: improving the description of alpha/gamma conformers. *Biophysical journal*, 92(11):3817–29, jun 2007.

[80] Mark J. Hackett, James A. McQuillan, Fatima El-Assaad, Jade B. Aitken, Aviva Levina, David D. Cohen, Rainer Siegele, Elizabeth A. Carter, Georges E. Grau, Nicholas H. Hunt, and Peter A. Lay. Chemical alterations to murine brain tissue induced by formalin fixation: implications for biospectroscopic imaging and mapping studies of disease pathogenesis. *Journal of the American Chemical Society*, 111(23):8551–8566, 1989.

[81] Norman L. Allinger. Conformational analysis. 130. MM2. A hydrocarbon force field utilizing V1 and V2 torsional terms. *Journal of the American Chemical Society*, 99(25):8127–8134, 1977.

[82] Scott J Weiner, Peter A Kollman, David A Case, U Chandra Singh, Caterina Ghio, Guliano Alagona, Salvatore Profeta, Paul Weiner, Giuliano Alagona, and Paul Weinerl. A new force field for molecular mechanical simulation of nucleic acids and proteins A New Force Field for Molecular Mechanical Simulation of Nucleic Acids and Proteins. *Journal of the American Chemical Society*, 106(3):765–784, feb 1984.

[83] Felix A Faber, Alexander Lindmaa, O Anatole von Lilienfeld, and Rickard Armiento. Machine Learning Energies of 2 Million Elpasolite (A B C 2 D 6) Crystals. *Physical Review Letters*, 117(13):135502, 2016.

[84] Rafael Gomez-Bombarelli, Jennifer N. Wei, David Duvenaud, Jose Miguel Hernandez-Lobato, Benjamin Sanchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alan Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018. PMID: 29532027.

[85] Matthias Rupp, Raghunathan Ramakrishnan, and O Anatole von Lilienfeld. Machine learning for quantum mechanical properties of atoms in molecules. *The Journal of Physical Chemistry Letters*, 6(16):3309–3313, 2015.

[86] Haoyan Huo and Matthias Rupp. Unified Representation for Machine Learning of Molecules and Crystals. *arXiv preprint arXiv:1704.06439*, 2017.

[87] Nongnuch Artrith and Alexander Urban. An implementation of artificial neural-network potentials for atomistic materials simulations: Performance for tio2. *Computational Materials Science*, 114:135–150, 2016.

[88] Nongnuch Artrith, Alexander Urban, and Gerbrand Ceder. Efficient and accurate machine-learning interpolation of atomic energies in compositions with many species. *Physical Review B*, 96(1):014112, 2017.

[89] Li Li, Thomas E. Baker, Steven R. White, and Kieron Burke. Pure density functional for strong correlation and the thermodynamic limit from machine learning. *Phys. Rev. B*, 94:245129, Dec 2016.

[90] Li Li, John C Snyder, Isabelle M Pelaschier, Jessica Huang, Uma-Naresh Niranjan, Paul Duncan, Matthias Rupp, Klaus-Robert Müller, and Kieron Burke. Understanding machine-learned density functionals. *International Journal of Quantum Chemistry*, 116(11):819–833, 2016.

[91] David Weininger. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, 1988.

[92] Sandip De, Albert P Bartók, Gábor Csányi, and Michele Ceriotti. Comparing molecules and solids across structural and alchemical space. *Physical Chemistry Chemical Physics*, 18(20):13754–13769, 2016.

[93] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel,

and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[94] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei a Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, feb 2015.

[95] Kristof T Schütt, Pieter-Jan Kindermans, Huziel E Sauceda, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Moleculenet: A continuous-filter convolutional neural network for modeling quantum interactions. *arXiv preprint arXiv:1706.08566*, 2017.

[96] G. Kresse and J. Hafner. Ab initio molecular dynamics for liquid metals. *Physical Review B*, 47(1):558–561, 1993.

[97] G. Kresse and J. Hafner. Ab initio molecular-dynamics simulation of the liquid-metal-amorphous-semiconductor transition in germanium. *Physical Review B*, 49(20):14251–14269, 1994.

[98] G Kresse and J Furthmüller. Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set. *Computational Materials Science*, 6(15):15–50, 1996.

[99] G. Kresse. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Physical Review B*, 54(16):11169–11186, 1996.

[100] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[101] Ian Dewancker, Michael McCourt, Scott Clark, Patrick Hayes, Alexandra Johnson, and George Ke. Evaluation system for a bayesian optimization service. *arXiv preprint*

arXiv:1605.06170, 2016.

[102] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[103] Kyle Mills, Kevin Ryczko, Iryna Luchak, Adam Domurad, Chris Beeler, and Isaac Tamblyn. Extensive deep neural networks for transferring small scale learning to large scale systems. *Chemical science*, 10(15):4129–4140, 2019.

[104] Tejas D Kulkarni, William F. Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2539–2547. Curran Associates, Inc., 2015.

[105] Jörg Behler. Neural network potential-energy surfaces in chemistry: a tool for large-scale simulations. *Physical Chemistry Chemical Physics*, 13:17930–17955, 2011.

[106] Sönke Lorenz, Axel Groß, and Matthias Scheffler. Representing high-dimensional potential-energy surfaces for reactions at surfaces by neural networks. *Chemical Physics Letters*, 395(4):210–215, 2004.

[107] Chris M. Handley and Paul L. A. Popelier. Potential energy surfaces fitted by artificial neural networks. *The Journal of Physical Chemistry A*, 114(10):3371–3383, 2010. PMID: 20131763.

[108] Roman M. Balabin and Ekaterina I. Lomakina. Neural network approach to quantum-chemistry data: Accurate prediction of density functional theory energies. *The Journal of Chemical Physics*, 131(7):074104, 2009.

[109] Changjian Xie, Xiaolei Zhu, David R. Yarkony, and Hua Guo. Permutation invariant polynomial neural network approach to fitting potential energy surfaces. iv. coupled diabatic potential energy matrices. *The Journal of Chemical Physics*, 149(14):144107, 2018.

[110] Li Li, Thomas E. Baker, Steven R. White, and Kieron Burke. Pure density functional for strong correlation and the thermodynamic limit from machine learning. *Physical Review B*, 94:245129, Dec 2016.

[111] Li Li, John C. Snyder, Isabelle M. Pelaschier, Jessica Huang, Uma-Naresh Niranjan, Paul Duncan, Matthias Rupp, Klaus-Robert Müller, and Kieron Burke. Understanding machine-learned density functionals. *International Journal of Quantum Chemistry*, 116(11):819–833, 2016.

[112] John C. Snyder, Matthias Rupp, Katja Hansen, Leo Blooston, Klaus-Robert Müller, and Kieron Burke. Orbital-free bond breaking via machine learning. *Journal of Chemical Physics*, 139:224104, 2013.

[113] Brian Kolb, Levi C. Lentz, and Alexie M. Kolpak. Discovering charge density functionals and structure-property relationships with prophet: A general framework for coupling machine learning and first-principles methods. *Scientific Reports*, 7(1), 4 2017.

[114] Kun Yao and John Parkhill. Kinetic energy of hydrocarbons as a function of electron density and convolutional neural networks. *Journal of Chemical Theory and Computation*, 12(3):1139–1147, 2016.

[115] Mark S. Hybertsen and Steven G. Louie. Electron correlation in semiconductors and insulators: Band gaps and quasiparticle energies. *Physical Review B*, 34:5390–5413, Oct 1986.

[116] Kyle Mills, Michael Spanner, and Isaac Tamblyn. Deep learning and the schrödinger equation. *Physical Review A*, 96:042113, Oct 2017.

[117] P. A. M. Dirac. Note on exchange phenomena in the thomas atom. *Mathematical Proceedings of the Cambridge Philosophical Society*, 26(3):376–385, 1930.

[118] m F Bloch. Bemerkung zur elektronentheorie des ferromagnetismus und der elektrischen leitfähigkeit. *Zeitschrift für Physik A Hadrons and Nuclei*, 57(7):545–555, 1929.

[119] S Pittalis, E Räsänen, N Helbig, and EKU Gross. Exchange-energy functionals for finite two-dimensional systems. *Physical Review B*, 76(23):235314, 2007.

[120] Xavier Andrade, David Strubbe, Umberto De Giovannini, Ask Hjorth Larsen, Micael JT Oliveira, Joseba Alberdi-Rodriguez, Alejandro Varas, Iris Theophilou, Nicole Helbig, Matthieu J Verstraete, et al. Real-space grids and the octopus code as tools for the development of new simulation approaches for electronic systems. *Physical Chemistry Chemical Physics*, 17(47):31371–31396, 2015.

[121] Xavier Andrade, Joseba Alberdi-Rodriguez, David A Strubbe, Micael JT Oliveira, Fernando Nogueira, Alberto Castro, Javier Muguerza, Agustin Arruabarrena, Steven G Louie, Alán Aspuru-Guzik, et al. Time-dependent density-functional theory in massively parallel computer architectures: the octopus project. *Journal of Physics: Condensed Matter*, 24(23):233202, 2012.

[122] Xavier Andrade and Alán Aspuru-Guzik. Real-space density functional theory on graphical processing units: computational approach and comparison to gaussian basis set methods. *Journal of Chemical Theory and Computation*, 9(10):4360–4373, 2013.

[123] Yuan Tang. Tf. learn: Tensorflow's high-level module for distributed machine learning. *arXiv preprint arXiv:1612.04251*, 2016.

[124] Yan Alexander Wang and Emily A Carter. Orbital-free kinetic-energy density functional theory. In *Theoretical Methods in Condensed Phase Chemistry*, pages 117–184. Springer, 2002.

[125] Vincent L Lignères and Emily A Carter. An introduction to orbital-free density functional theory. In *Handbook of Materials Modeling*, pages 137–148. Springer, 2005.

[126] Linda Hung and Emily A Carter. Accurate simulations of metals at the mesoscale: Explicit treatment of 1 million atoms with quantum mechanics. *Chemical Physics Letters*, 475(4-6):163–170, 2009.

[127] Paul Gombas. Handbuch der physik. *Springer, Berlin*, 36:109–231, 1956.

[128] NH March. The thomas-fermi approximation in quantum mechanics. *Advances in Physics*, 6(21):1–101, 1957.

[129] Elliott H Lieb. Erratum: Thomas-fermi and related theories of atoms and molecules. *Reviews of Modern Physics*, 54(1):311, 1982.

[130] BI Dunlap, JWD Connolly, and JR Sabin. On first-row diatomic molecules and local density models. *The Journal of Chemical Physics*, 71(12):4993–4999, 1979.

[131] Ralf Meyer, Manuel Weichselbaum, and Andreas W. Hauser. Machine learning approaches toward orbital-free density functional theory: Simultaneous training on the kinetic energy density functional and its functional derivative. *Journal of Chemical Theory and Computation*, 16(9):5685–5694, 2020. PMID: 32786898.

[132] Ryo Nagai, Ryosuke Akashi, and Osamu Sugino. Completing density functional theory by machine learning hidden messages from molecules. *NPJ Computational Materials*, 6(1):1–8, 2020.

[133] Bhupalee Kalita, Li Li, Ryan J McCarty, and Kieron Burke. Learning to approximate density functionals. *Accounts of Chemical Research*, 54(4):818–826, 2021.

[134] Alexander Ryabov, Iskander Akhatov, and Petr Zhilyaev. Neural network interpolation of exchange-correlation functional. *Scientific Reports*, 10(1):1–7, 2020.

[135] Keenan Lyon. *From Fundamentals to Spectroscopic Applications of Density Functional Theory*. PhD thesis, 2020.

[136] Chao Yang, Juan C Meza, and Lin-Wang Wang. A constrained optimization algorithm for total energy minimization in electronic structure calculations. *Journal of Computational Physics*, 217(2):709–721, 2006.

[137] Valery Weber, Joost VandeVondele, Juerg Hutter, and Anders MN Niklasson. Direct energy functional minimization under orthogonality constraints. *The Journal of Chemical Physics*, 128(8):084113, 2008.

[138] Xuecheng Shao, Kaili Jiang, Wenhui Mi, Alessandro Genova, and Michele Pavanello. Dftpy: An efficient and object-oriented platform for orbital-free dft simulations. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 11(1):e1482, 2021.

[139] MJ Van Setten, Matteo Giantomassi, Eric Bousquet, Matthieu J Verstraete, Don R Hamann, Xavier Gonze, and G-M Rignanese. The pseudodojo: Training and grading a 85 element optimized norm-conserving pseudopotential table. *Computer Physics Communications*, 226:39–54, 2018.

[140] Xavier Gonze, Bernard Amadon, Gabriel Antonius, Frederic Arnardi, Lucas Baguet, Jean-Michel Beuken, Jordan Bieder, Francois Bottin, Johann Bouchet, Eric Bousquet, Nils Brouwer, Fabien Bruneval, Guillaume Brunin, Theo Cavignac, Jean-Baptiste Charraud, Wei Chen, Michel Cote, Stefaan Cottenier, Jules Denier, Gregory Geneste, Philippe Ghosez, Matteo Giantomassi, Yannick Gillet, Olivier Gingras, Donald R. Hamann, Geoffroy Hautier, Xu He, Nicole Helbig, Natalie Holzwarth, Yongchao Jia, Francois Jollet, William Lafargue-Dit-Hauret, Kurt Lejaeghere, Miguel A. L. Marques, Alexandre Martin, Cyril Martins, Henrique P. C. Miranda, Francesco Naccarato, Kristin Persson, Guido Petretto, Valentin Planes, Yann Pouillon, Sergei Prokhorenko, Fabio Ricci, Gian-Marco Rignanese, Aldo H. Romero, Michael Marcus Schmitt, Marc Torrent, Michiel J. van Setten, Benoit Van Troeye, Matthieu J. Verstraete, Gilles ZÃl'rah, and Josef W. Zwanziger. The abinit project: Impact, environment and recent developments. *Comput. Phys. Commun.*, 248:107042, 2020.

[141] Shubin Liu and Paul W Ayers. Functional derivative of noninteracting kinetic energy density functional. *Physical Review A*, 70(2):022501, 2004.

[142] Kevin Ryczko, Kyle Mills, Iryna Luchak, Christa Homenick, and Isaac Tamblyn. Convolutional neural networks for atomistic systems. *Computational Materials Science*, 149:134–142, 2018.

[143] Kevin Ryczko, David A Strubbe, and Isaac Tamblyn. Deep learning and density-functional theory. *Physical Review A*, 100(2):022512, 2019.

[144] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *arXiv preprint arXiv:1706.02515*, 2017.

[145] Kevin Ryczko and Isaac Tamblyn. *Codes*, 2021. clean.energyscience.ca/codes.

[146] S Alireza Ghasemi and Thomas D Kühne. Artificial neural networks for the kinetic energy functional of non-interacting fermions. *The Journal of Chemical Physics*, 154(7):074107, 2021.

[147] O Anatole von Lilienfeld and Kieron Burke. Retrospective on a decade of machine learning for chemical discovery. *Nature Communications*, 11(1):1–4, 2020.

[148] Tim Mueller, Alberto Hernandez, and Chuhong Wang. Machine learning for interatomic potential models. *The Journal of Chemical Physics*, 152(5):050902, 2020.

[149] Andrea Grisafi, David M Wilkins, Gábor Csányi, and Michele Ceriotti. Symmetry-adapted machine learning for tensorial properties of atomistic systems. *Physical Review Letters*, 120(3):036002, 2018.

[150] David M Wilkins, Andrea Grisafi, Yang Yang, Ka Un Lao, Robert A DiStasio, and Michele Ceriotti. Accurate molecular polarizabilities with coupled cluster theory and machine learning. *Proceedings of the National Academy of Sciences*, 116(9):3401–3406, 2019.

[151] Anders S Christensen, Felix A Faber, and O Anatole von Lilienfeld. Operators in quantum machine learning: Response properties in chemical space. *The Journal of chemical physics*, 150(6):064105, 2019.

[152] Hasan Babaei, Ruiqiang Guo, Amirreza Hashemi, and Sangyeop Lee. Machine-learning-based interatomic potential for phonon transport in perfect crystalline si and crystalline si with vacancies. *Physical Review Materials*, 3(7):074603, 2019.

[153] Andrew Rohskopf, Spencer Wyant, Kiarash Gordiz, Hamid Reza Seyf, Murali Gopal Muraleedharan, and Asegun Henry. Fast & accurate interatomic potentials for describing thermal vibrations. *Computational Materials Science*, 184:109884, 2020.

[154] Bohayra Mortazavi, Ivan S Novikov, Evgeny V Podryabinkin, Stephan Roche, Timon Rabczuk, Alexander V Shapeev, and Xiaoying Zhuang. Exploring phononic properties of two-dimensional materials using machine learning interatomic potentials. *Applied Materials Today*, 20:100685, 2020.

[155] Nathaniel Raimbault, Andrea Grisafi, Michele Ceriotti, and Mariana Rossi. Using gaussian process regression to simulate the vibrational raman spectra of molecular crystals. *New Journal of Physics*, 21(10):105001, 2019.

[156] Tsz Wai Ko, Jonas A Finkler, Stefan Goedecker, and Jörg Behler. General-purpose machine learning potentials capturing nonlocal charge transfer. *Accounts of Chemical Research*, 54(4):808–817, 2021.

[157] RD King-Smith and David Vanderbilt. Theory of polarization of crystalline solids. *Physical Review B*, 47(3):1651, 1993.

[158] Na Sai, Karin M Rabe, and David Vanderbilt. Theory of structural response to macroscopic electric fields in ferroelectric systems. *Physical Review B*, 66(10):104108, 2002.

[159] Xavier Gonze and Changyol Lee. Dynamical matrices, born effective charges, dielectric permittivity tensors, and interatomic force constants from density-functional perturbation theory. *Physical Review B*, 55(16):10355, 1997.

[160] Aldo H Romero, Douglas C Allan, Bernard Amadon, Gabriel Antonius, Thomas Applencourt, Lucas Baguet, Jordan Bieder, Francois Bottin, Johann Bouchet, Eric Bousquet, et al. Abinit: Overview and focus on selected capabilities. *The Journal of Chemical Physics*, 152(12):124102, 2020.

[161] Razvan Caracas and Ronald E Cohen. Theoretical determination of the raman spectra of mgsio3 perovskite and post-perovskite at high pressure. *Geophysical Research Letters*, 33(12), 2006.

[162] Sandro Mignuzzi, Andrew J Pollard, Nicola Bonini, Barry Brennan, Ian S Gilmore, Marcos A Pimenta, David Richards, and Debdulal Roy. Effect of disorder on raman scattering of single-layer mo s 2. *Physical Review B*, 91(19):195411, 2015.

[163] Xavier Gonze, Bernard Amadon, Gabriel Antonius, Frederic Arnardi, Lucas Baguet, Jean-Michel Beuken, Jordan Bieder, Francois Bottin, Johann Bouchet, Eric Bousquet, et al. The abinit project: Impact, environment and recent developments. *Computer Physics Communications*, 248:107042, 2020.

[164] Xavier Gonze. First-principles responses of solids to atomic displacements and homogeneous electric fields: Implementation of a conjugate-gradient algorithm. *Phys. Rev. B*, 55(16):10337–10354, April 1997.

[165] Kevin Ryczko, Sebastian J Wetzel, Roger G Melko, and Isaac Tamblyn. Orbital-free density functional theory with small datasets and deep learning. *arXiv preprint arXiv:2104.05408*, 2021.

[166] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

[167] O Malenfant-Thuot, K Ryczko, M Côté, and I Tamblyn. Training deep neural networks for derivative predictions in materials science. *In preparation.*, 2021.

[168] Jonathan Laflamme Janssen, Michel Côté, Steven G Louie, and Marvin L Cohen. Electron-phonon coupling in c 60 using hybrid functionals. *Physical Review B*, 81(7):073106, 2010.

[169] Marc OJ Jäger, Eiaki V Morooka, Filippo Federici Canova, Lauri Himanen, and Adam S Foster. Machine learning hydrogen adsorption on nanoclusters through structural descriptors. *NPJ Computational Materials*, 4(1):1–8, 2018.

[170] Felix Faber, Alexander Lindmaa, O Anatole von Lilienfeld, and Rickard Armiento. Crystal structure representations for machine learning models of formation energies. *International Journal of Quantum Chemistry*, 115(16):1094–1101, 2015.

[171] Benjamin G Peyton, Connor Briggs, Ruhee D'Cunha, Johannes T Margraf, and T Daniel Crawford. Machine-learning coupled cluster properties through a density tensor representation. *The Journal of Physical Chemistry A*, 124(23):4861–4871, 2020.

[172] Johannes T Margraf and Karsten Reuter. Making the coupled cluster correlation energy machine-learnable. *The Journal of Physical Chemistry A*, 122(30):6343–6348, 2018.

[173] Christoph Schran, Jörg Behler, and Dominik Marx. Automated fitting of neural network potentials at coupled cluster accuracy: Protonated water clusters as testing ground. *Journal of Chemical Theory and Computation*, 16(1):88–99, 2019.

[174] Mihail Bogojeski, Leslie Vogt-Maranto, Mark E Tuckerman, Klaus-Robert Müller, and Kieron Burke. Quantum chemical accuracy from density functional approximations via machine learning. *Nature Communications*, 11(1):1–11, 2020.

[175] Jaron T Krogel, Min Yu, Jeongnim Kim, and David M Ceperley. Quantum energy density: Improved efficiency for quantum monte carlo calculations. *Physical Review B*, 88(3):035137, 2013.

[176] M Burkatzki, C Filippi, and M Dolg. Energy-consistent pseudopotentials for quantum monte carlo calculations. *The Journal of Chemical Physics*, 126(23):234105, 2007.

[177] Paolo Giannozzi, Stefano Baroni, Nicola Bonini, Matteo Calandra, Roberto Car, Carlo Cavazzoni, Davide Ceresoli, Guido L Chiarotti, Matteo Cococcioni, Ismaila Dabo, et al. Quantum espresso: a modular and open-source software project for quantum simulations of materials. *Journal of physics: Condensed matter*, 21(39):395502, 2009.

[178] Jeongnim Kim, Andrew D Baczewski, Todd D Beaudet, Anouar Benali, M Chandler Bennett, Mark A Berrill, Nick S Blunt, Edgar Josué Landinez Borda, Michele Casula, David M Ceperley, et al. Qmcpack: an open source ab initio quantum monte carlo package for the electronic structure of atoms, molecules and solids. *Journal of Physics: Condensed Matter*, 30(19):195901, 2018.

[179] Jaron T Krogel. Nexus: A modular workflow management system for quantum simulation codes. *Computer Physics Communications*, 198:154–168, 2016.

[180] L Li, S Reich, and J Robertson. Defect energies of graphite: Density-functional calculations. *Physical Review B*, 72(18):184109, 2005.

[181] Justin S Smith, Ben Nebgen, Nicholas Lubbers, Olexandr Isayev, and Adrian E Roitberg. Less is more: Sampling chemical space with active learning. *The Journal of Chemical Physics*, 148(24):241733, 2018.

[182] Kevin Ryczko and Isaac Tamblyn. *Datasets*, 2021. http://clean.energyscience.ca/datasets.

[183] SD Bader and SSP Parkin. Spintronics. *Annu. Rev. Condens. Matter Phys.*, 1(1):71–88, 2010.

[184] John R Schaibley, Hongyi Yu, Genevieve Clark, Pasqual Rivera, Jason S Ross, Kyle L Seyler, Wang Yao, and Xiaodong Xu. Valleytronics in 2d materials. *Nature Reviews Materials*, 1(11):16055, 2016.

[185] Kin Fai Mak, Keliang He, Jie Shan, and Tony F Heinz. Control of valley polarization in monolayer MoS2 by optical helicity. *Nature Nanotechnology*, 7:494, 6 2012.

[186] Hualing Zeng, Junfeng Dai, Wang Yao, Di Xiao, and Xiaodong Cui. Valley polarization in MoS2 monolayers by optical pumping. *Nature Nanotechnology*, 7:490, 6 2012.

[187] Ting Cao, Gang Wang, Wenpeng Han, Huiqi Ye, Chuanrui Zhu, Junren Shi, Qian Niu, Pingheng Tan, Enge Wang, Baoli Liu, and Ji Feng. Valley-selective circular dichroism of monolayer molybdenum disulphide. *Nature Communications*, 3:887, 6 2012.

[188] K F Mak, K L McGill, J Park, and P L McEuen. The valley Hall effect in MoS2 transistors. *Science*, 344(6191):1489–1492, 2014.

[189] A H Castro Neto, F Guinea, N M R Peres, K S Novoselov, and A K Geim. The electronic properties of graphene. *Reviews of Modern Physics*, 81(1):109–162, 1 2009.

[190] A Rycerz, J Tworzydło, and C W J Beenakker. Valley filter and valley valve in graphene. *Nature Physics*, 3:172, 2 2007.

[191] Di Xiao, Wang Yao, and Qian Niu. Valley-contrasting physics in graphene: magnetic moment and topological transport. *Physical Review Letters*, 99(23):236809, 2007.

[192] Thomas Stegmann and Nikodem Szpak. Current splitting and valley polarization in elastically deformed graphene. *2D Materials*, 6(1):015024, 2018.

[193] Gareth W Jones, Dario Andres Bahamon, Antonio H Castro Neto, and Vitor M Pereira. Quantized transport, strain-induced perfectly conducting modes, and valley filtering on shape-optimized graphene corbino devices. *Nano letters*, 17(9):5304–5313, 2017.

[194] N Levy, SA Burke, KL Meaker, M Panlasigui, A Zettl, F Guinea, AH Castro Neto, and MF Crommie. Strain-induced pseudo–magnetic fields greater than 300 tesla in graphene nanobubbles. *Science*, 329(5991):544–547, 2010.

[195] F Guinea, Baruch Horovitz, and P Le Doussal. Gauge field induced by ripples in graphene. *Physical Review B*, 77(20):205421, 2008.

[196] Mikkel Settnes, Stephen R Power, and Antti-Pekka Jauho. Pseudomagnetic fields and triaxial strain in graphene. *Physical Review B*, 93(3):035456, 2016.

[197] Mikkel Settnes, Stephen R Power, Mads Brandbyge, and Antti-Pekka Jauho. Graphene nanobubbles as valley filters and beam splitters. *Physical Review Letters*, 117(27):276801, 2016.

[198] Dawei Zhai and Nancy Sandler. Local versus extended deformed graphene geometries for valley filtering. *Physical Review B*, 98(16):165437, 2018.

[199] Thomas Stegmann and Nikodem Szpak. Current flow paths in deformed graphene: from quantum transport to classical trajectories in curved space. *New Journal of Physics*, 18(5):053016, 2016.

[200] Yong Wu, D Zhai, C Pan, B Cheng, T Taniguchi, K Watanabe, N Sandler, and M Bockrath. Quantum wires and waveguides formed in graphene by strain. *Nano letters*, 18(1):64–69, 2017.

[201] Elias Andrade, Ramon Carrillo-Bastos, and Gerardo G Naumis. Valley engineering by strain in kekulé-distorted graphene. *Physical Review B*, 99(3):035411, 2019.

[202] Yasuhiro Hatsugai. So small implies so large: For a material design. *JPSJ News and Comments*, 16:13, 2019.

[203] AC McRae, Guoqing Wei, and AR Champagne. Graphene quantum strain transistors. *Physical Review Applied*, 11(5):054019, 2019.

[204] T Fujita, MBA Jalil, and SG Tan. Valley filter in strain engineered graphene. *Applied Physics Letters*, 97(4):043508, 2010.

[205] Daniel Gunlycke and Carter T White. Graphene valley filter using a line defect. *Physical Review Letters*, 106(13):136806, 2011.

[206] J-H Chen, G Autès, Nasim Alem, F Gargiulo, A Gautam, M Linck, C Kisielowski, OV Yazyev, SG Louie, and A Zettl. Controlled growth of a line defect in graphene and implications for gate-tunable valley filtering. *Physical Review B*, 89(12):121407, 2014.

[207] Vadim V Cheianov, Vladimir Falko, and B L Altshuler. The Focusing of Electron Flow and a Veselago Lens in Graphene p-n Junctions. *Science*, 315(5816):1252–1255, 2007.

[208] Changsoo Park. Magnetoelectrically controlled valley filter and valley valve in bilayer graphene. *Physical Review Applied*, 11(4):044033, 2019.

[209] JL Garcia-Pomar, A Cortijo, and M Nieto-Vesperinas. Fully valley-polarized electron beams in graphene. *Physical Review Letters*, 100(23):236801, 2008.

[210] Thomas Aktor, Jose H Garcia, Stephan Roche, Antti-Pekka Jauho, and Stephen R Power. Topological valley currents in graphene with local sublattice asymmetry. *arXiv preprint arXiv:1910.00489*, 2019.

[211] Amir Natan, Leeor Kronik, Hossam Haick, and Raymond T Tung. Electrostatic properties of ideal and non-ideal polar organic monolayers: Implications for electronic devices. *Advanced Materials*, 19(23):4103–4117, 2007.

[212] Christoph W Groth, Michael Wimmer, Anton R Akhmerov, and Xavier Waintal. Kwant: a software package for quantum transport. *New Journal of Physics*, 16(6):63065, 2014.

[213] Sebastian Steiger, Michael Povolotskyi, Hong-Hyun Park, Tillmann Kubis, and Gerhard Klimeck. Nemo5: A parallel multiscale nanoelectronics modeling tool. *IEEE Transactions on Nanotechnology*, 10(6):1464–1474, 2011.

[214] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[215] Alexander Y Piggott, Jesse Lu, Konstantinos G Lagoudakis, Jan Petykiewicz, Thomas M Babinec, and Jelena Vučković. Inverse design and demonstration of a compact and broadband on-chip wavelength demultiplexer. *Nature Photonics*, 9:374, 5 2015.

[216] Logan Su, Alexander Y Piggott, Neil V Sapra, Jan Petykiewicz, and Jelena Vučković. Inverse design and demonstration of a compact on-chip narrowband three-channel wavelength demultiplexer. *ACS Photonics*, 5(2):301–305, 2018.

[217] Riccardo Leardi. Genetic algorithms in chemometrics and chemistry: a review. *Journal of Chemometrics*, 15(7):559–569, 2001.

[218] V Torres, P Silva, EAT de Souza, LA Silva, and DA Bahamon. Valley notch filter in a graphene strain superlattice: Green's function and machine learning approach. *Physical Review B*, 100(20):205411, 2019.

[219] Daniele Melati, Yuri Grinberg, Mohsen Kamandar Dezfouli, Siegfried Janz, Pavel Cheben, Jens H Schmid, Alejandro Sánchez-Postigo, and Dan-Xia Xu. Mapping the global design space of nanophotonic components using machine learning pattern recognition. *Nature Communications*, 10(1):1–9, 2019.

[220] Thomas C Schelling. Dynamic models of segregation. *The Journal of Mathematical Sociology*, 1(2):143–186, 1971.

[221] Stephen Wolfram. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55(3):601, 1983.

[222] Katsunori Wakabayashi and Takashi Aoki. Electrical conductance of zigzag nanographite ribbons with locally applied gate voltage. *International Journal of Modern Physics B*, 16(32):4897–4909, 2002.

[223] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2224–2232. Curran Associates, Inc., 2015.

[224] Chris Beeler, Uladzimir Yahorau, Rory Coles, Kyle Mills, Stephen Whitelam, and Isaac Tamblyn. Optimizing thermodynamic trajectories using evolutionary reinforcement learning. *arXiv preprint arXiv:1903.08543*, 2019.

[225] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*, 2017.

[226] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.